Making Programming Synonymous with Parallel Programming for Linear Algebra

Robert A. van de Geijn

Department of Computer Sciences

The University of Texas at Austin

The Current Core FLAME Team

- UT-Austin
  - Ernie Chan
  - Victor Eijkhout
  - Kazushige Goto
  - Field Van Zee
  - (this list is a bit out of date)
- UJI-Spain
  - Enrique Quintana-Orti
  - Gregorio Quintana-Orti
- Aachen University
  - Paolo Bientinesi
- Others
  - See http://www.cs.utexas.edu/users/flame

The presented work is also in collaboration with

- Maribel Castillo
- Francisco D. Igual
- Rafael Mayo

(Univ. Jaume I, Spain)

Funding

- This project is primarily supported by multiple grants from NSF
- Unrestricted grants from NEC, Dr. Truchard (CEO of National Instruments)
- For a little $$$ UTK's or DOE's name could be here…

What is FLAME

- A notation for representing algorithms
- A methodology for deriving correct algorithms
- APIs for representing algorithms in code
- A library (libFLAME)
- A pedagogical tool

In other words:

FLAME is a programmer's lifestyle choice

The Need for Forward Compatibility

- An operation is encountered for which no library routine exists
- A novel architecture comes along
- The traditional library does not deliver the performance one wants
- A novel data storage scheme comes along

What we need is a flexible infrastructure

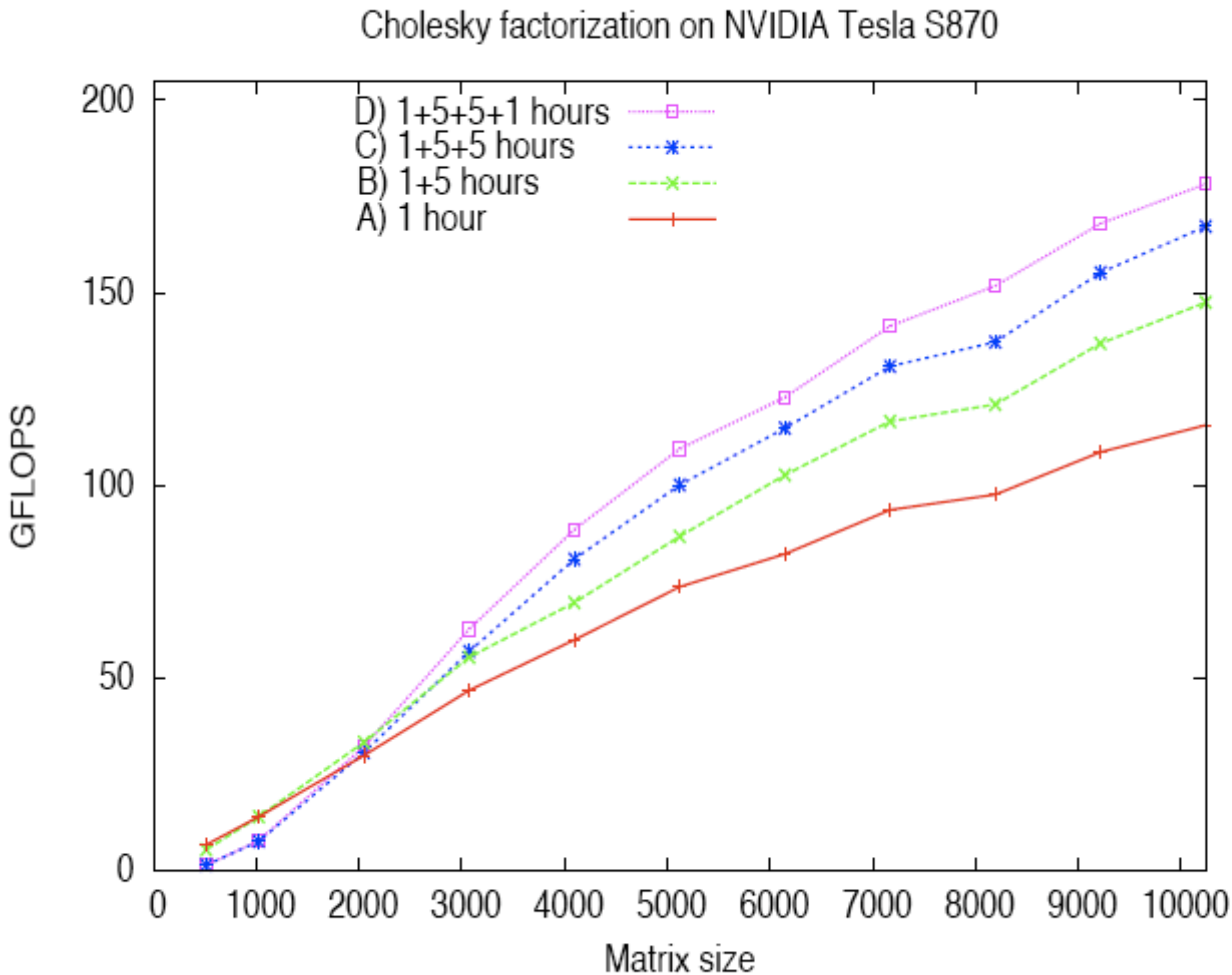
Motivating scenario

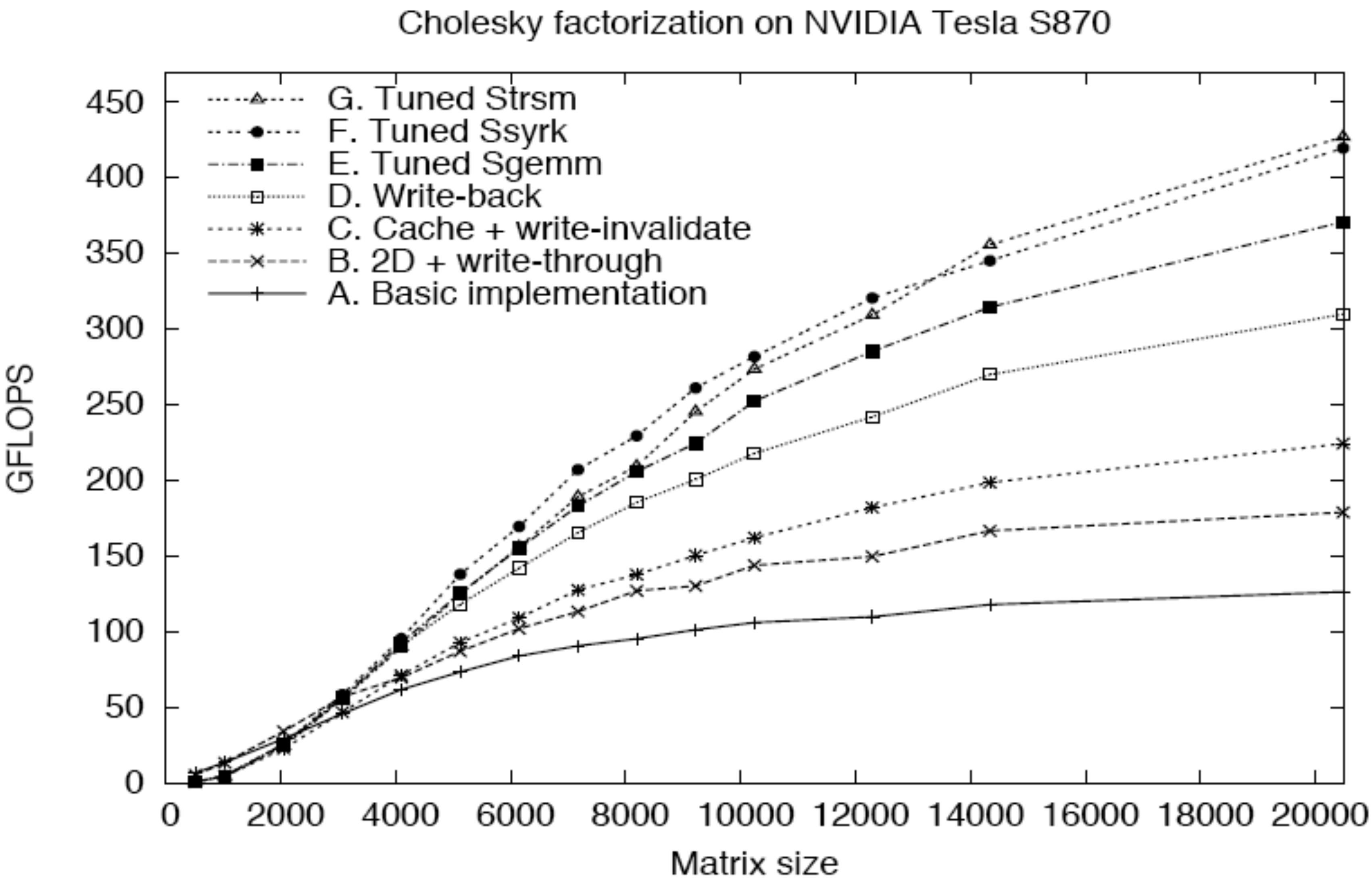
- A new architecture comes along. How much effort does it take to port a typical library routine to this architecture?
  - NVIDIA Tesla S870 system
    - 4 NVIDIA G80 GPUs
    - 6 Gbytes DDR3 memory
    - Peak 1.4 Tflops single precision
    - Available libraries: sequential CUBLAS 1.1 (later 2.0)
    - Host: Intel Xeon QuadCore E5405 (2.0 GHz)
      - Two PCIExpress Gen2 interfaces
        (peak bandwidth: 48 Gbytes/sec per interface)

What do we have to do?

- View the matrix as a collection of blocks that are stored contiguously
  - This is (thought to be) hard
- Program the algorithm as an algorithm-by-blocks
  - This is (thought to be) hard
- Schedule the operations with blocks and manage the movement of blocks between the host and the GPUs
  - This is (thought to be) hard

Welcome to programming hell


Cholesky factorization on NVIDIA Tesla S870
D) 1+5+5+1 hours
C) 1+5+5 hours
B) 1+5 hours
A) 1 hour
GFLOPS
200
150
100
50
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Matrix size



Cholesky factorization on NVIDIA Tesla S870
G. Tuned Strsm
F. Tuned Ssyrk
E. Tuned Sgemm
D. Write-back
C. Cache + write-invalidate
B. 2D + write-through
A. Basic implementation
GFLOPS
Matrix size
0
50
100
150
200
250
300
350
400
450
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000

Overview

- A bit of history
- Programming algorithms-by-blocks is ~~hard~~ easy
  - An example: Cholesky factorization by blocks
- Programming algorithms-by-blocks ~~further~~ simplifies ~~complicates~~ programming (future) multicore architectures
- Conclusion
- Resources

Overview

- A bit of history
- Programming algorithms-by-blocks is ~~hard~~ easy
  - An example: Cholesky factorization by blocks
- Programming algorithms-by-blocks ~~further~~ simplifies ~~complicates~~ programming (future) multicore architectures
- Conclusion
- Resources

A Bit of History

Once upon a time there was package…

- 1972: EISPACK
  - Package for the solution of the dense eigenvalue problem
    - E.g.: $A x = \lambda x$
  - First robust software pack
    - Numerical stability, performance, and portability were a concern
    - Consistent formatting of code
  - Coded in Algol
  - First released in 1972

Basic Linear Algebra Subprograms (BLAS)

- In the 1970s vector supercomputers ruled
- Dense (and many sparse) linear algebra algorithms can be formulated in terms of vector operations
- By standardizing an interface to such operations, portable high performance could be achieved
  - Vendors responsible for optimized implementations
  - Other libraries coded in terms of the BLAS
- First proposed in 1973. Published in

  C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh,
  Basic Linear Algebra Subprograms for FORTRAN usage,
  ACM Trans. Math. Soft., 5 (1979).
- Later became known as the level-1 BLAS
- Fortran77 interface

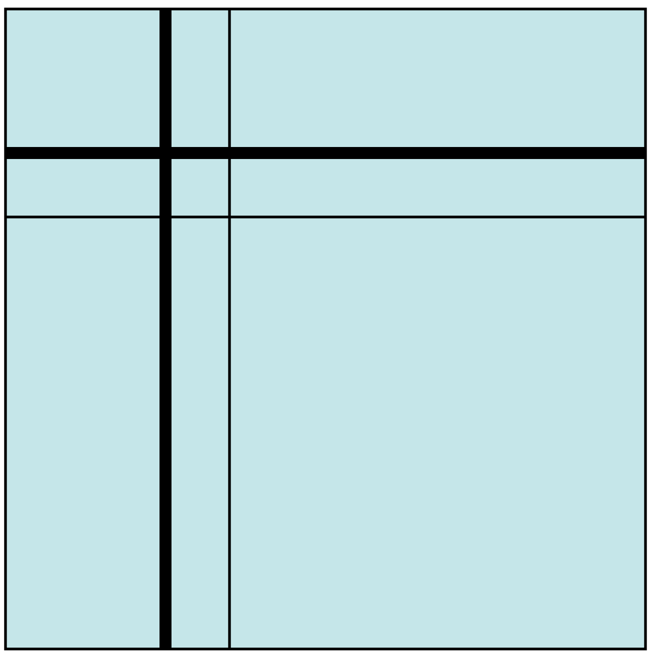
Linear Algebra Package (LINPACK)

- Targeted solution of linear equations and linear least-squares
- Coded in terms of the level-1 BLAS for portability
- Started in 1974. Published/released in 1977

  J. J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart. LINPACK User's Guide, SIAM, 1977

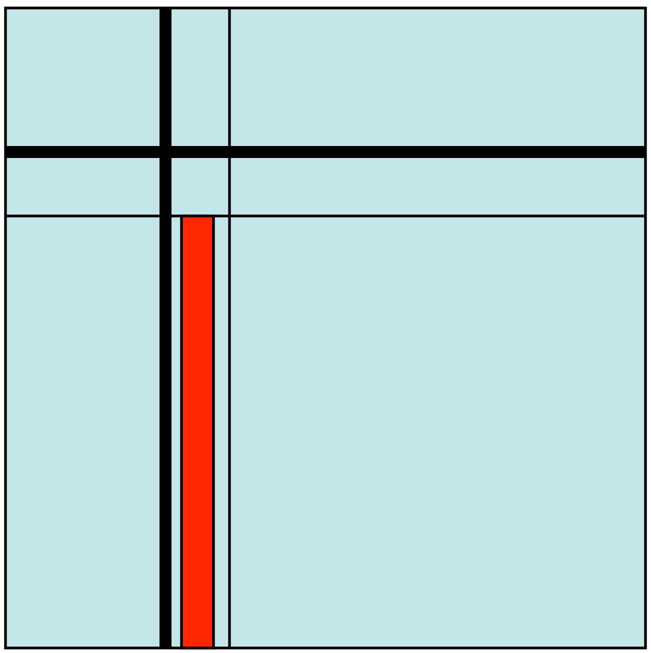
- Fortran66 interface

LINPACK
LU factorization
with partial pivoting
(abbreviated)



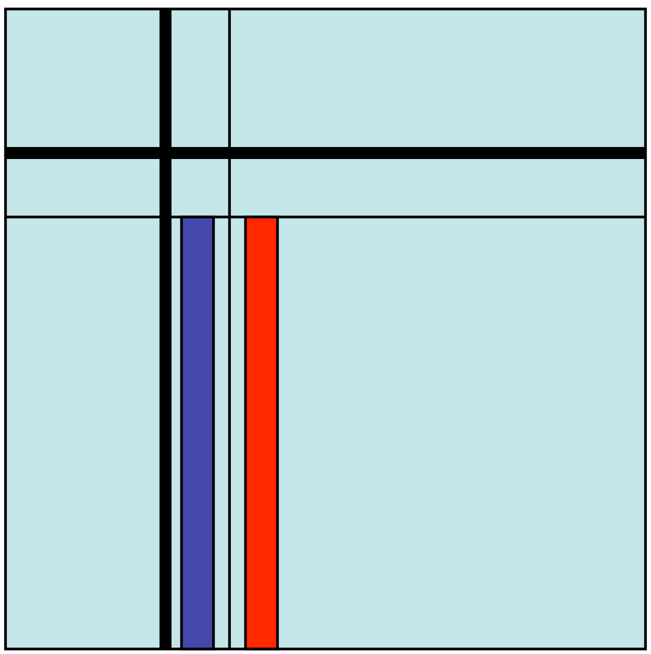
```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK
LU factorization
with partial pivoting
(abbreviated)



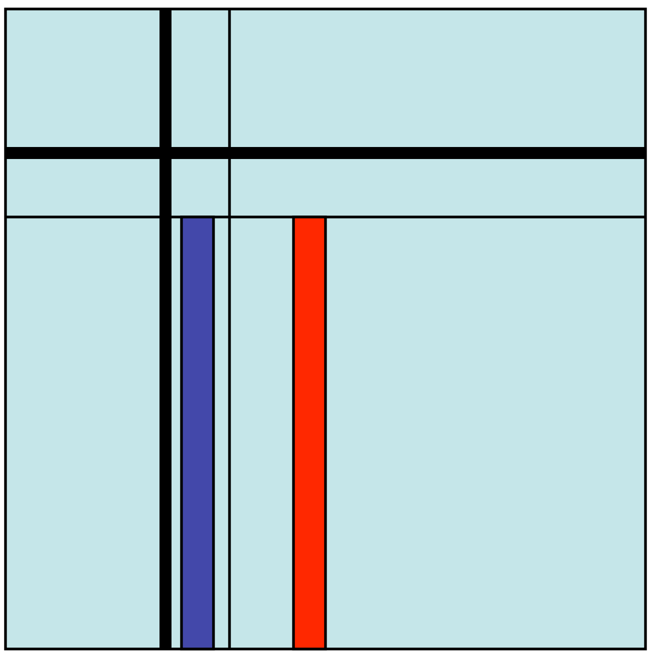
```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK
LU factorization
with partial pivoting
(abbreviated)



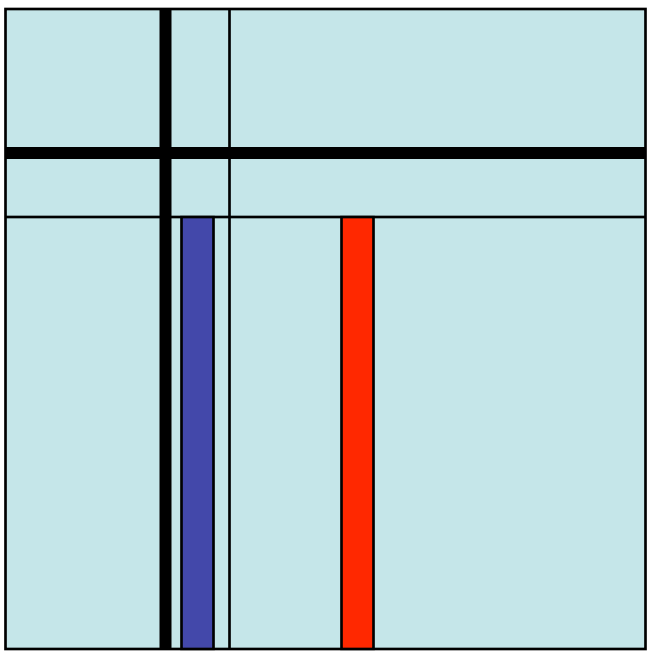
```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK
LU factorization
with partial pivoting
(abbreviated)



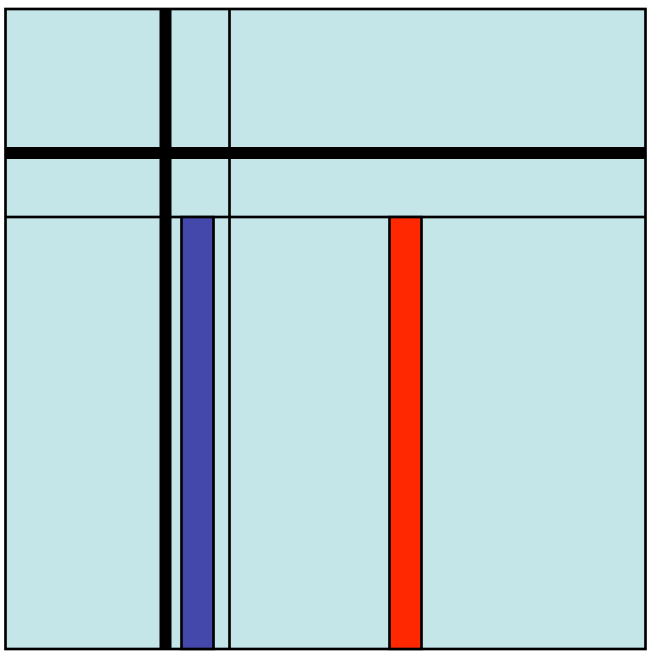
```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK
LU factorization
with partial pivoting
(abbreviated)



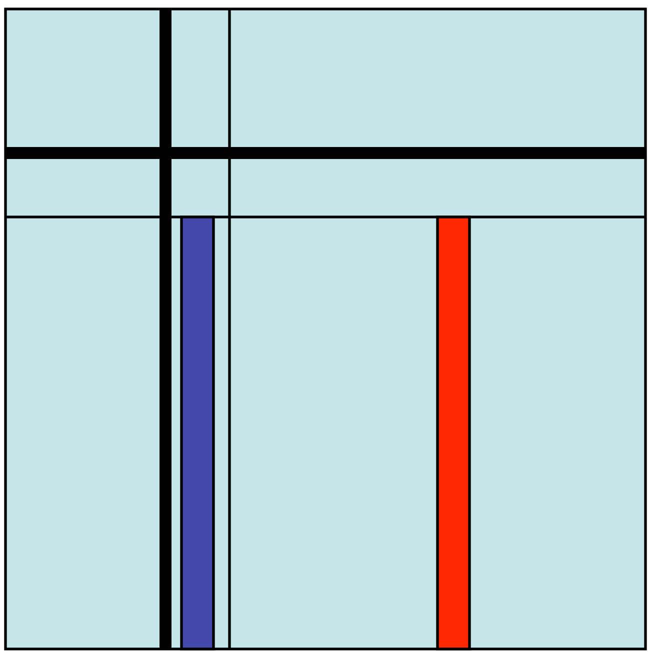
```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK
LU factorization
with partial pivoting
(abbreviated)



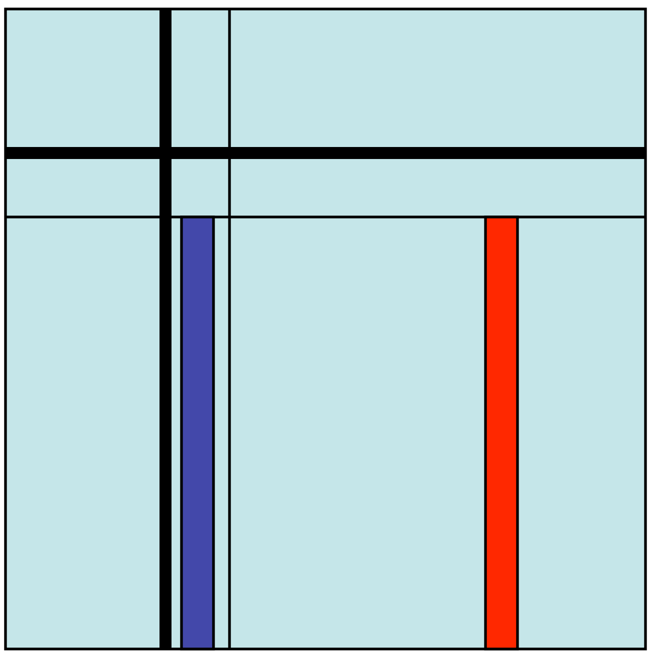
```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK
LU factorization
with partial pivoting
(abbreviated)



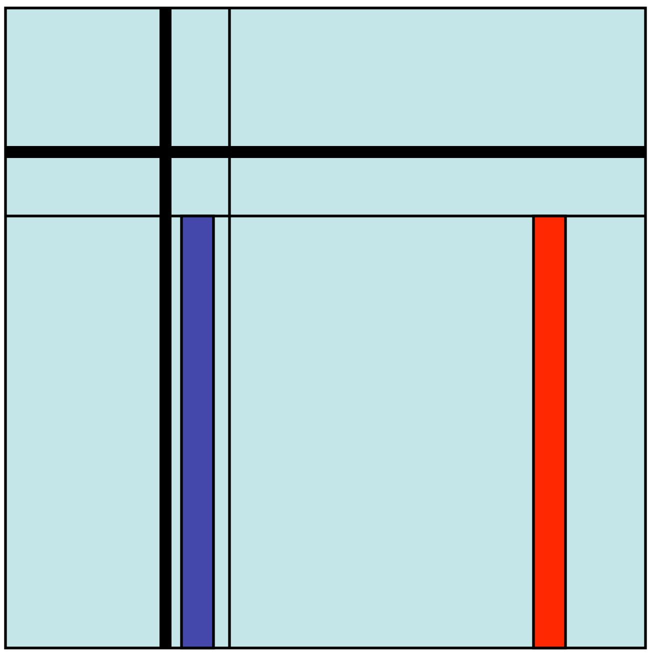
```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK
LU factorization
with partial pivoting
(abbreviated)



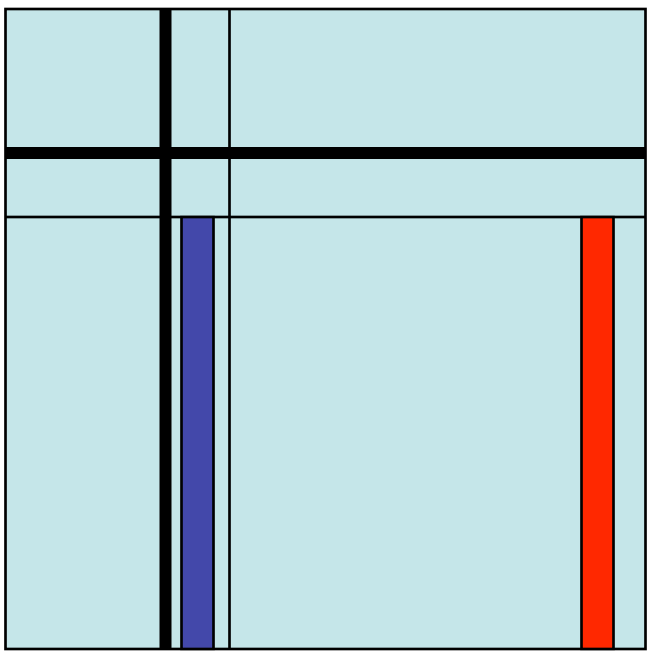
```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK
LU factorization
with partial pivoting
(abbreviated)



```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK
LU factorization
with partial pivoting
(abbreviated)

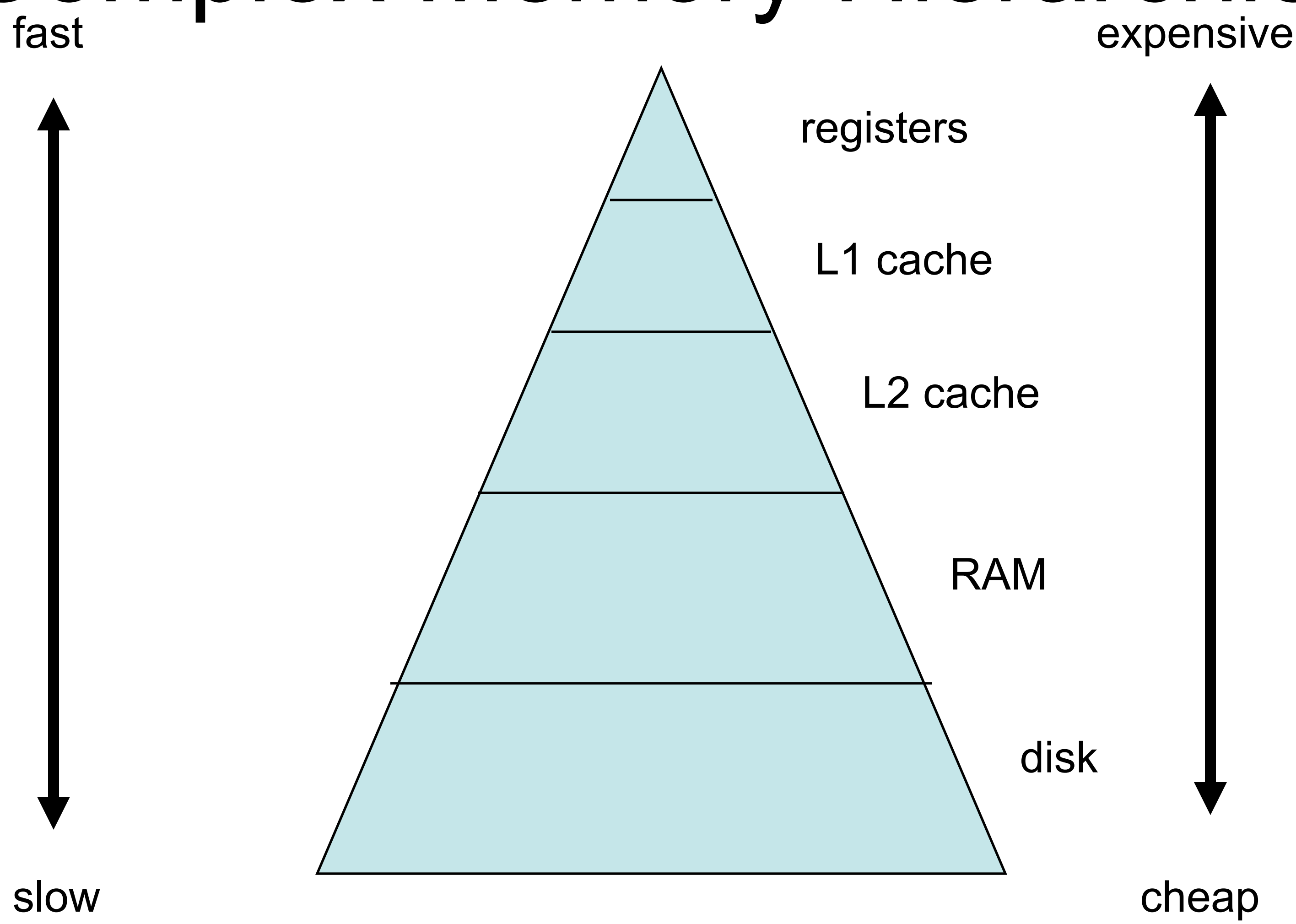


```
      do 60 k = 1, nm1
         kp1 = k + 1

         l = idamax(n-k+1,a(k,k),1) + k - 1
         ipvt(k) = l

         if (a(l,k) .eq. 0.0d0) go to 40

            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

            t = -1.0d0/a(k,k)
            call dscal(n-k,t,a(k+1,k),1)

            row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call daxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
```

LINPACK (continued)

- Portable high performance on vector architectures
- Poor performance when architectures with complex memory hierarchies arrived in the 1980s
  - O(n) operations on O(n) data means bandwidth to main memory becomes the limiting factor

Complex Memory Hierarchies

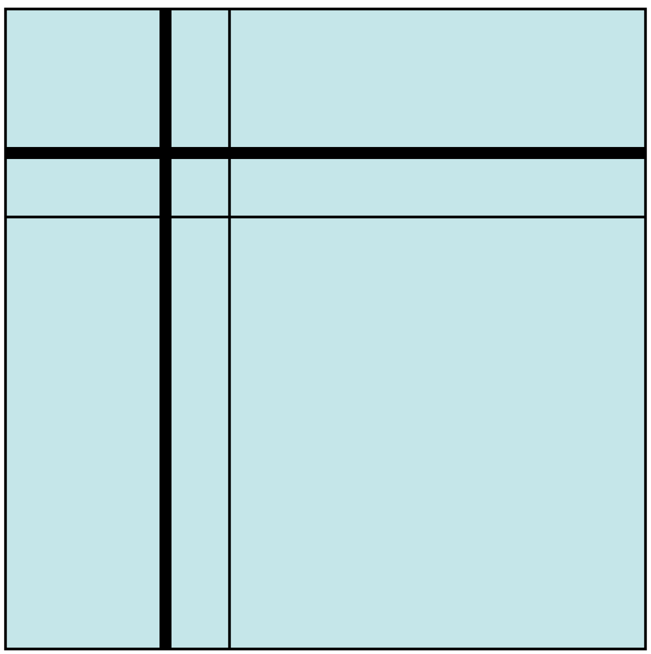


Evolving Towards Higher Performance

- Level 2 BLAS: matrix-vector operations
  - Started in 1984. Published in 1988

    J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, *An extended set of FORTRAN Basic Linear Algebra Subprograms*, ACM Trans. Math. Soft., 14 (1988), pp. 1--17.
  - Casts computation in terms of operations like matrix-vector multiplication and rank-1 update
  - Benefit: vector(s) can be kept in cache memory
  - Problem: O(n^2) operations on O(n^2) data

LAPACK
LU factorization
with partial pivoting
unblocked algorithm
(abbreviated)



```
      DO 10 J = 1, MIN( M, N )

         JP = J - 1 + IDAMAX( M-J+1, A( J, J ), 1 )
         IPIV( J ) = JP
         IF( A( JP, J ).NE.ZERO ) THEN

            IF( JP.NE.J )
     $         CALL DSWAP( N, A( J, 1 ), LDA, A( JP, 1 ), LDA )

            IF( J.LT.M ) THEN
               IF( ABS(A( J, J )) .GE. SFMIN ) THEN
                  CALL DSCAL( M-J, ONE / A( J, J ), A( J+1, J ), 1 )
               ELSE
                 DO 20 I = 1, M-J
                    A( J+I, J ) = A( J+I, J ) / A( J, J )
   20            CONTINUE
               END IF
            END IF

         ELSE IF( INFO.EQ.0 ) THEN

            INFO = J
         END IF

         IF( J.LT.MIN( M, N ) ) THEN

            Update trailing submatrix.

            CALL DGER( M-J, N-J, -ONE, A( J+1, J ), 1, A( J, J+1 ), LDA,
     $                 A( J+1, J+1 ), LDA )
         END IF
   10 CONTINUE
```

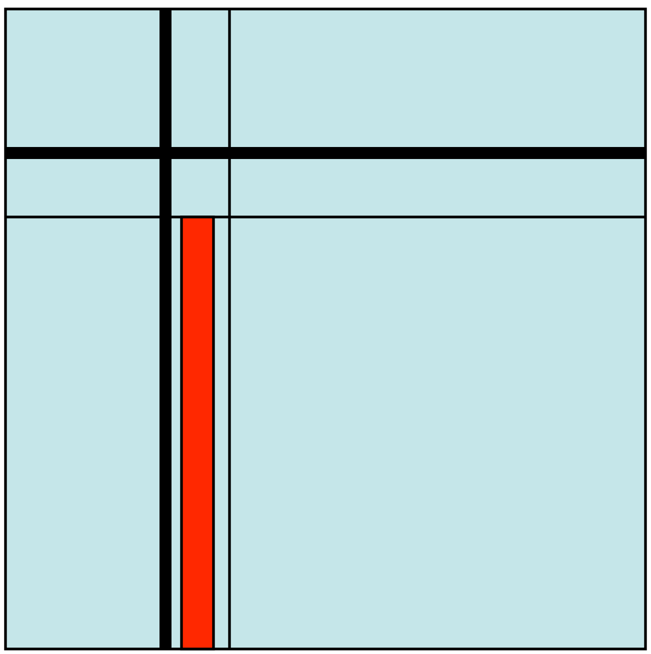
LAPACK

LU factorization

with partial pivoting

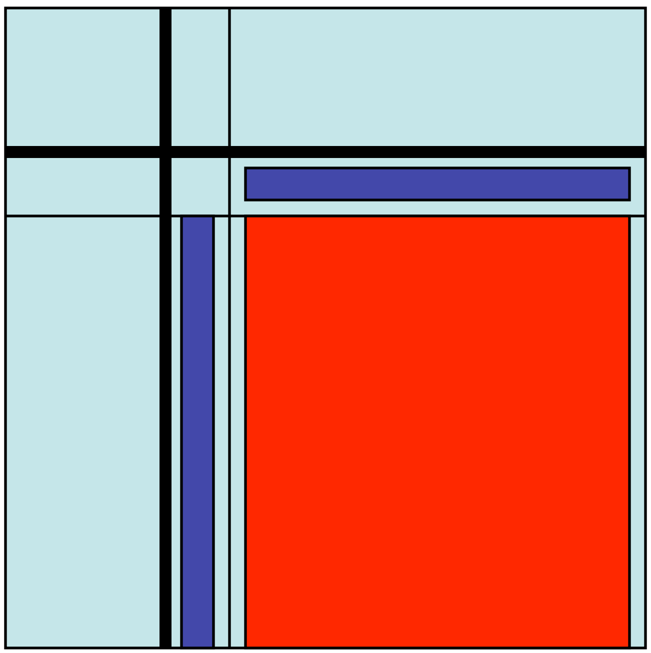
unblocked algorithm

(abbreviated)



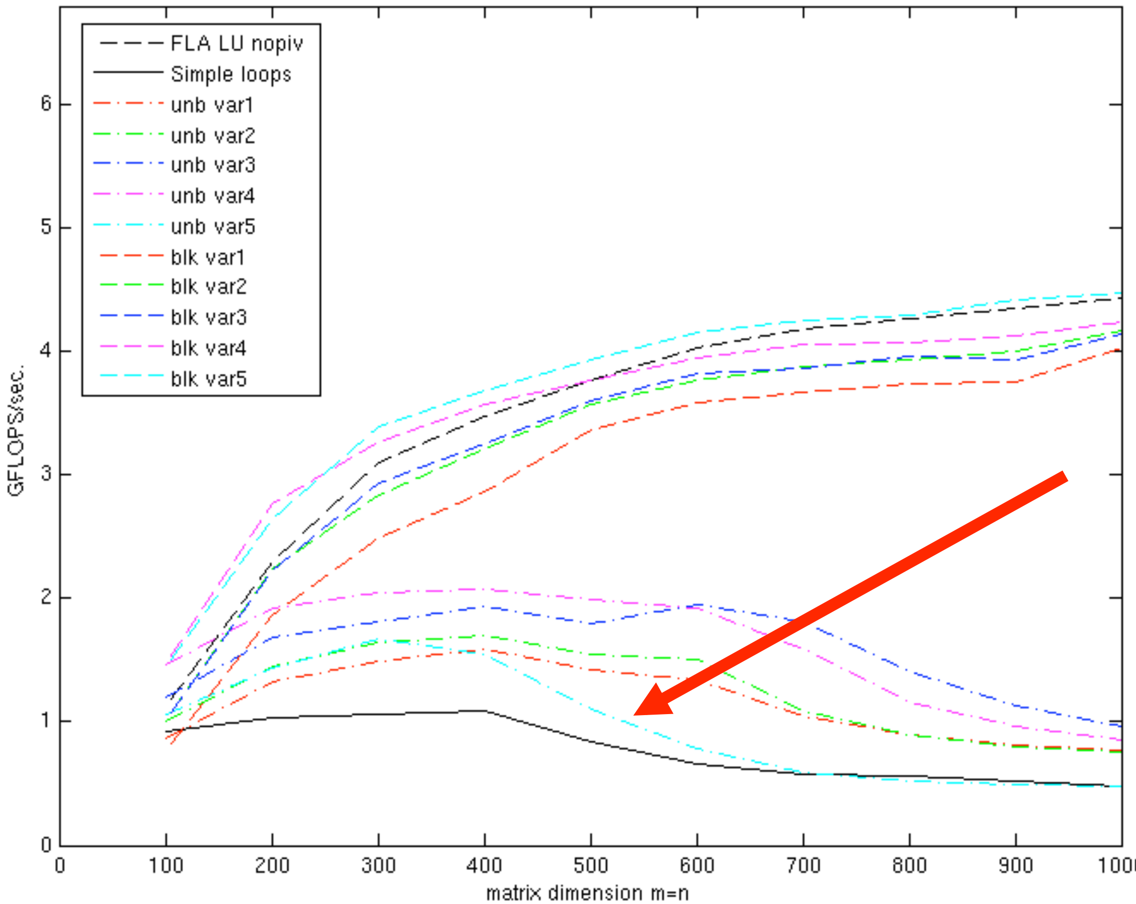
```
   DO 10 J = 1, MIN( M, N )

      JP = J - 1 + IDAMAX( M-J+1, A( J, J ), 1 )
      IPIV( J ) = JP
      IF( A( JP, J ).NE.ZERO ) THEN

         IF( JP.NE.J )
  $         CALL DSWAP( N, A( J, 1 ), LDA, A( JP, 1 ), LDA )

         IF( J.LT.M ) THEN
            IF( ABS(A( J, J )) .GE. SFMIN ) THEN
               CALL DSCAL( M-J, ONE / A( J, J ), A( J+1, J ), 1 )
            ELSE
              DO 20 I = 1, M-J
                 A( J+I, J ) = A( J+I, J ) / A( J, J )
20            CONTINUE
            END IF
         END IF

      ELSE IF( INFO.EQ.0 ) THEN

         INFO = J
      END IF

      IF( J.LT.MIN( M, N ) ) THEN

         Update trailing submatrix.

         CALL DGER( M-J, N-J, -ONE, A( J+1, J ), 1, A( J, J+1 ), LDA,
  $                 A( J+1, J+1 ), LDA )
      END IF
10 CONTINUE
```

LAPACK
LU factorization
with partial pivoting
unblocked algorithm
(abbreviated)



```
      DO 10 J = 1, MIN( M, N )

         JP = J - 1 + IDAMAX( M-J+1, A( J, J ), 1 )
         IPIV( J ) = JP
         IF( A( JP, J ).NE.ZERO ) THEN

            IF( JP.NE.J )
     $         CALL DSWAP( N, A( J, 1 ), LDA, A( JP, 1 ), LDA )

            IF( J.LT.M ) THEN
               IF( ABS(A( J, J )) .GE. SFMIN ) THEN
                  CALL DSCAL( M-J, ONE / A( J, J ), A( J+1, J ), 1 )
               ELSE
                 DO 20 I = 1, M-J
                    A( J+I, J ) = A( J+I, J ) / A( J, J )
   20            CONTINUE
               END IF
            END IF

         ELSE IF( INFO.EQ.0 ) THEN

            INFO = J
         END IF

         IF( J.LT.MIN( M, N ) ) THEN

            Update trailing submatrix.

            CALL DGER( M-J, N-J, -ONE, A( J+1, J ), 1, A( J, J+1 ), LDA,
     $                 A( J+1, J+1 ), LDA )
         END IF
   10 CONTINUE
```

Performance (Intel Xeon 3.4GHz)



Evolving Towards High Performance

- Level 3 BLAS: matrix-matrix operations
  - Started in 1986. Published in 1990

    J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, *A set of Level 3 Basic Linear Algebra Subprograms*, ACM Trans. Math. Soft., 16 (1990)

  - Casts computation in terms of operations like matrix-matrix multiplication
  - Benefit: submatrices can be kept in cache
  - O(n^3) operations on O(n^2) data

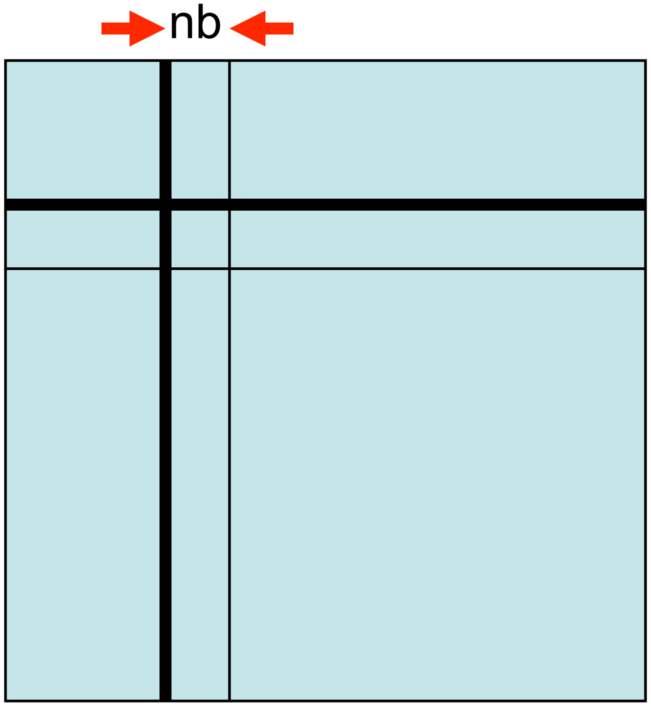
LAPACK

LU factorization

with partial pivoting

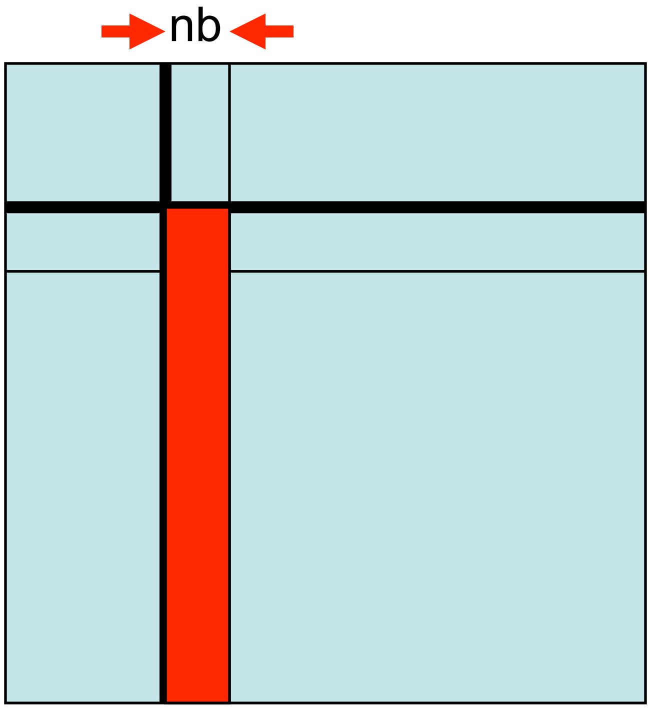
blocked algorithm

(abbreviated)




```
      DO 20 J = 1, MIN( M, N ), NB
         JB = MIN( MIN( M, N )-J+1, NB )

         CALL DGETF2( M-J+1, JB, A( J, J ), LDA, IPIV( J ), IINFO )

         IF( INFO.EQ.0 .AND. IINFO.GT.0 )
     $      INFO = IINFO + J - 1
         DO 10 I = J, MIN( M, J+JB-1 )
            IPIV( I ) = J - 1 + IPIV( I )
   10    CONTINUE

         CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )

         IF( J+JB.LE.N ) THEN
            CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,
     $                   IPIV, 1 )
            CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB,
     $                  N-J-JB+1, ONE, A( J, J ), LDA, A( J, J+JB ),
     $                  LDA )
            IF( J+JB.LE.M ) THEN
               CALL DGEMM( 'No transpose', 'No transpose', M-J-JB+1,
     $                     N-J-JB+1, JB, -ONE, A( J+JB, J ), LDA,
     $                     A( J, J+JB ), LDA, ONE, A( J+JB, J+JB ),
     $                     LDA )
            END IF
         END IF
   20 CONTINUE
```

LAPACK
LU factorization
with partial pivoting
blocked algorithm
(abbreviated)




```
      DO 20 J = 1, MIN( M, N ), NB
         JB = MIN( MIN( M, N )-J+1, NB )

         CALL DGETF2( M-J+1, JB, A( J, J ), LDA, IPIV( J ), IINFO )

         IF( INFO.EQ.0 .AND. IINFO.GT.0 )
     $      INFO = IINFO + J - 1
         DO 10 I = J, MIN( M, J+JB-1 )
            IPIV( I ) = J - 1 + IPIV( I )
   10    CONTINUE

         CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )

         IF( J+JB.LE.N ) THEN
            CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,
     $                   IPIV, 1 )
            CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB,
     $                  N-J-JB+1, ONE, A( J, J ), LDA, A( J, J+JB ),
     $                  LDA )
            IF( J+JB.LE.M ) THEN
               CALL DGEMM( 'No transpose', 'No transpose', M-J-JB+1,
     $                     N-J-JB+1, JB, -ONE, A( J+JB, J ), LDA,
     $                     A( J, J+JB ), LDA, ONE, A( J+JB, J+JB ),
     $                     LDA )
            END IF
         END IF
   20 CONTINUE
```

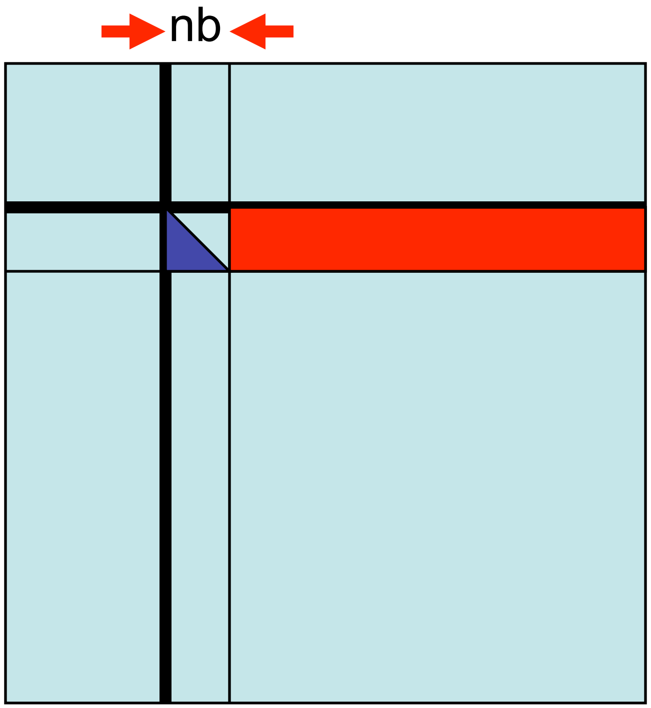
LAPACK

LU factorization

with partial pivoting

blocked algorithm

(abbreviated)




```
      DO 20 J = 1, MIN( M, N ), NB
         JB = MIN( MIN( M, N )-J+1, NB )

         CALL DGETF2( M-J+1, JB, A( J, J ), LDA, IPIV( J ), IINFO )

         IF( INFO.EQ.0 .AND. IINFO.GT.0 )
    $       INFO = IINFO + J - 1
         DO 10 I = J, MIN( M, J+JB-1 )
            IPIV( I ) = J - 1 + IPIV( I )
   10    CONTINUE

         CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )

         IF( J+JB.LE.N ) THEN
            CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,
    $                    IPIV, 1 )
            CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB,
    $                   N-J-JB+1, ONE, A( J, J ), LDA, A( J, J+JB ),
    $                   LDA )
            IF( J+JB.LE.M ) THEN
               CALL DGEMM( 'No transpose', 'No transpose', M-J-JB+1,
    $                      N-J-JB+1, JB, -ONE, A( J+JB, J ), LDA,
    $                      A( J, J+JB ), LDA, ONE, A( J+JB, J+JB ),
    $                      LDA )
            END IF
         END IF
   20 CONTINUE
```

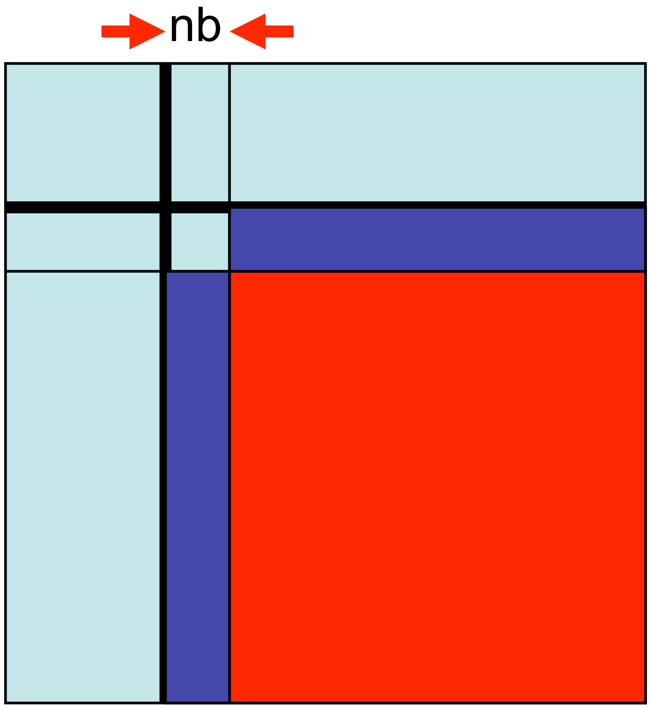
LAPACK

LU factorization

with partial pivoting

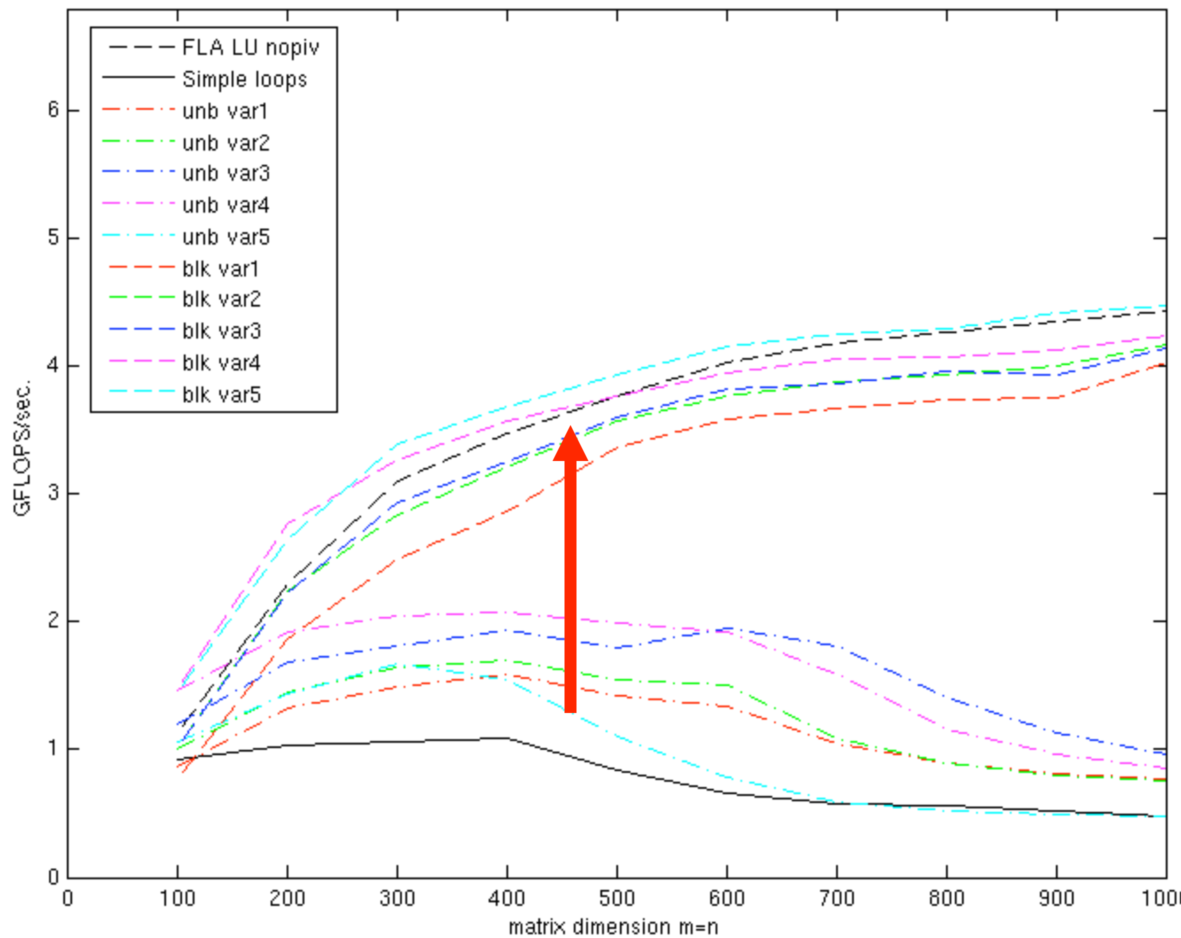
blocked algorithm

(abbreviated)




```
         DO 20 J = 1, MIN( M, N ), NB
            JB = MIN( MIN( M, N )-J+1, NB )

            CALL DGETF2( M-J+1, JB, A( J, J ), LDA, IPIV( J ), IINFO )

            IF( INFO.EQ.0 .AND. IINFO.GT.0 )
     $         INFO = IINFO + J - 1
            DO 10 I = J, MIN( M, J+JB-1 )
               IPIV( I ) = J - 1 + IPIV( I )
   10       CONTINUE

            CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )

            IF( J+JB.LE.N ) THEN
               CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1,
     $                      IPIV, 1 )
               CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB,
     $                     N-J-JB+1, ONE, A( J, J ), LDA, A( J, J+JB ),
     $                     LDA )
               IF( J+JB.LE.M ) THEN
                  CALL DGEMM( 'No transpose', 'No transpose', M-J-JB+1,
     $                        N-J-JB+1, JB, -ONE, A( J+JB, J ), LDA,
     $                        A( J, J+JB ), LDA, ONE, A( J+JB, J+JB ),
     $                        LDA )
               END IF
            END IF
   20    CONTINUE
```

Performance (Intel Xeon 3.4GHz)

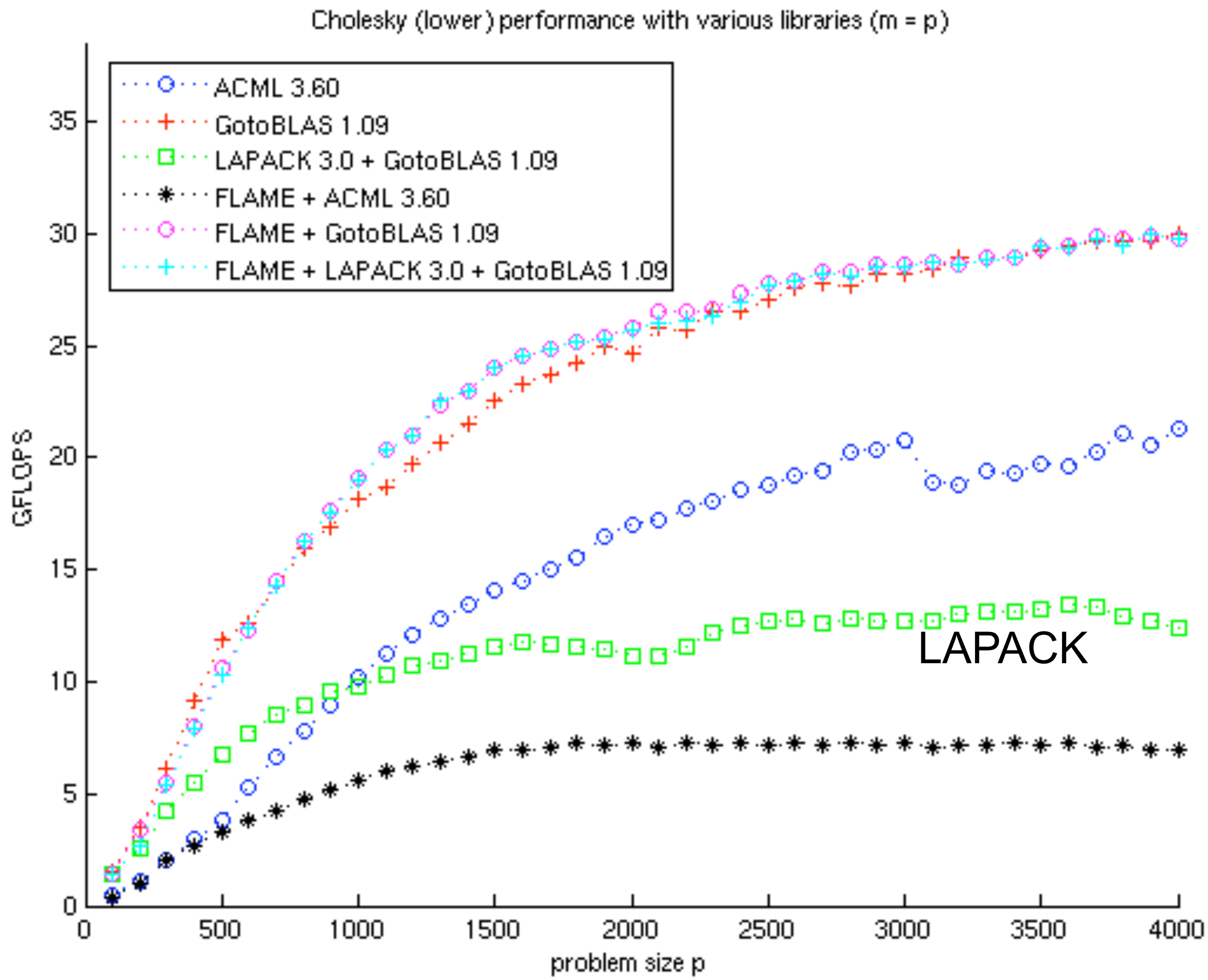


Evolving Beyond?

- New architectures with many “cores”
  - Symmetric Multi-Processors (SMPs)
  - Multicore architectures

- Algorithms encoded in LAPACK don’t always parallelize well

AMD Opteron 8 cores (2.4GHz)



Evolution vs Intelligent Design

- LAPACK code is hard to write/read/maintain/alter
- Formal Linear Algebra Methods Environment (FLAME) Project
  - Based on insights from the PLAPACK project
  - Started around 2000. First official library release (libFLAME 1.0): April 1, 2007
  - Collaboration between UT Dept. of Computer Sciences, TACC, and UJI-Spain
  - Systematic approach to deriving, presenting, and implementing algorithms

Why the FLAME API?

- From the “LAPACK 3.1.1. changes” log:

replaced calls of the form

```
CALL SCOPY( N, WORK, 1, TAU, 1 )
```

with

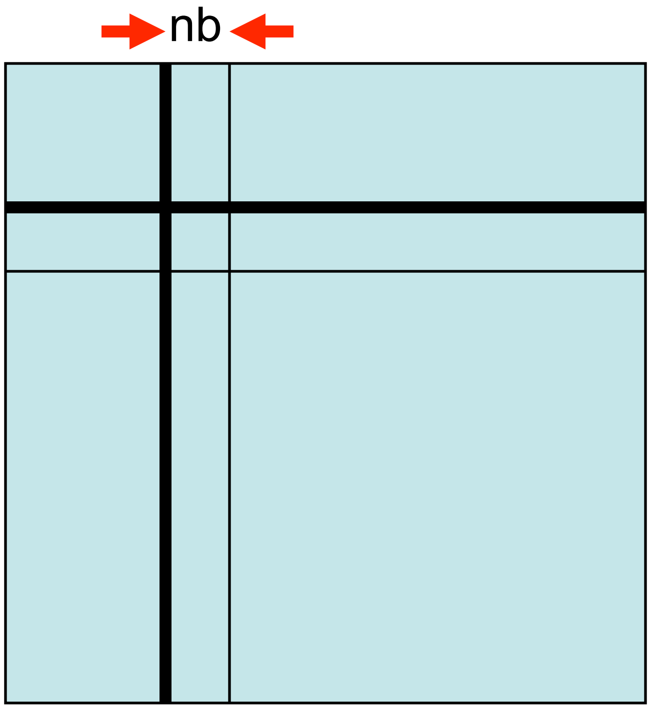
```
CALL SCOPY( N-1, WORK, 1, TAU, 1 )
```

at line 694 for s/dchkhs and line 698 for c/zchkhs.
(TAU is only of length N-1.)

FLAME

LU factorization

blocked algorithm



Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

  where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

  Determine block size b

  Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

  where A_{11} is $b \times b$

$A_{11} = \text{LU}(A_{11})$

$A_{12} = \text{TRILU}(A_{11})^{-1} A_{12}$

$A_{21} = A_{21} \text{TRIU}(A_{11})^{-1}$

$A_{22} = A_{22} - A_{21} A_{12}$

  Continue with

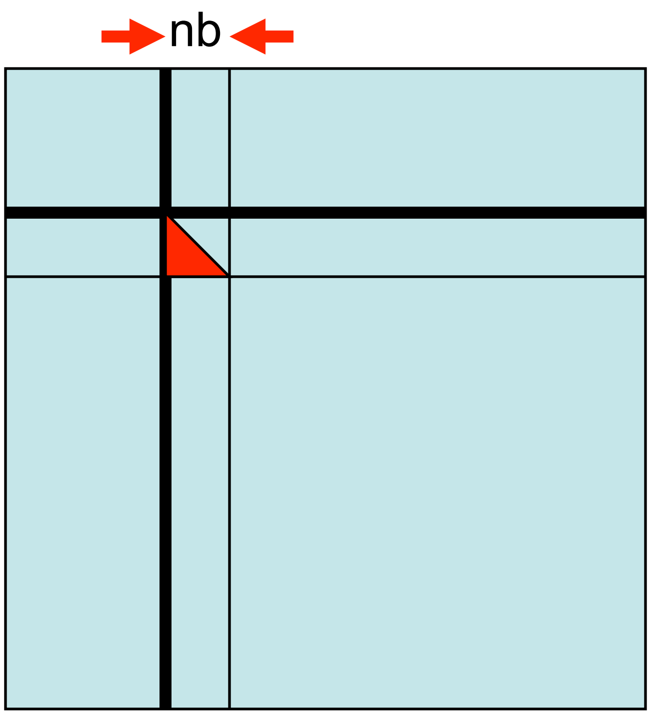
$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

FLAME

LU factorization

blocked algorithm



Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$\boxed{A_{11} = \text{LU}(A_{11})}$

$A_{12} = \text{TRILU}(A_{11})^{-1} A_{12}$

$A_{21} = A_{21} \text{TRIU}(A_{11})^{-1}$

$A_{22} = A_{22} - A_{21} A_{12}$

Continue with

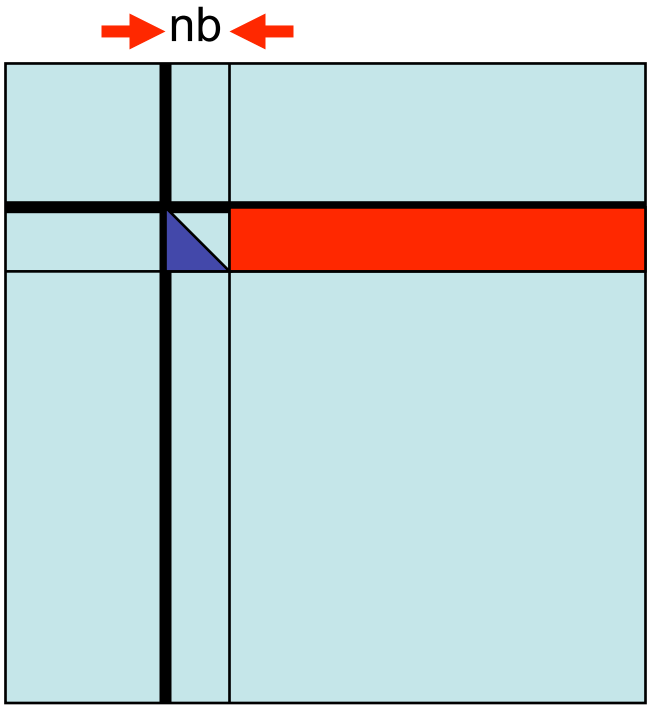
$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

FLAME

LU factorization

blocked algorithm



Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} = \text{LU}(A_{11})$

$\boxed{A_{12} = \text{TRILU}(A_{11})^{-1} A_{12}}$

$A_{21} = A_{21}\,\text{TRIU}(A_{11})^{-1}$

$A_{22} = A_{22} - A_{21} A_{12}$

Continue with

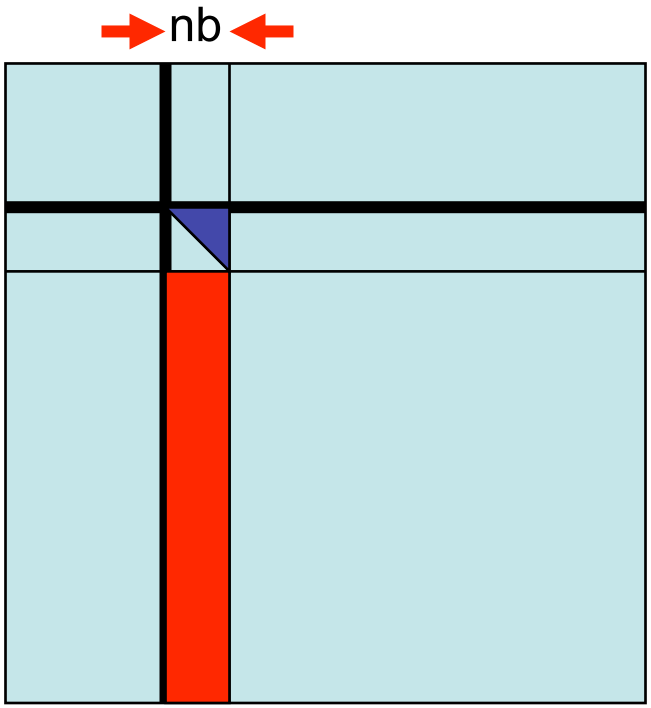
$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

FLAME

LU factorization

blocked algorithm



Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} = \text{LU}(A_{11})$

$A_{12} = \text{TRILU}(A_{11})^{-1} A_{12}$

$\boxed{A_{21} = A_{21}\,\text{TRIU}(A_{11})^{-1}}$

$A_{22} = A_{22} - A_{21} A_{12}$

Continue with

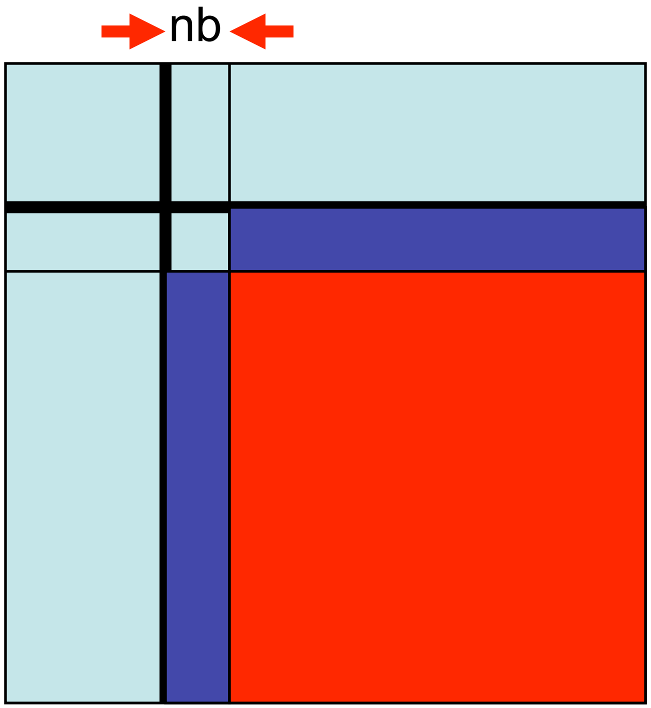
$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

FLAME

LU factorization

blocked algorithm



Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} = \text{LU}(A_{11})$

$A_{12} = \text{TRILU}(A_{11})^{-1} A_{12}$

$A_{21} = A_{21} \text{TRIU}(A_{11})^{-1}$

$\boxed{A_{22} = A_{22} - A_{21} A_{12}}$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$$A_{11} = \text{LU}(A_{11})$$
$$A_{12} = \text{TRILU}(A_{11})^{-1} A_{12}$$
$$A_{21} = A_{21}\,\text{TRIU}(A_{11})^{-1}$$
$$A_{22} = A_{22} - A_{21} A_{12}$$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

```
FLA_Part_2x2( A,    &ATL, &ATR,
                     &ABL, &ABR,     0, 0, FLA_TL );

 while (FLA_Obj_length( ATL ) < FLA_Obj_length( A )){

   b = min( FLA_Obj_length( ABR ), nb_alg );

   FLA_Repart_2x2_to_3x3
      ( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
     /* ************* */   /* ****************** */
                             &A10, /**/ &A11, &A12,
        ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
        b, b, FLA_BR );
   /*------------------------------------------------*/
   LU_unb_var5( A11 );

   FLA_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
             FLA_NO_TRANSPOSE, FLA_UNIT_DIAG,
             FLA_ONE, A11, A12 );

   FLA_Trsm( FLA_RIGHT, FLA_UPPER_TRIANGULAR,
             FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
             FLA_ONE, A11, A21 );

   FLA_Gemm( FLA_NO_TRANSPOSE, FLA_NO_TRANSPOSE,
             FLA_MINUS_ONE, A21, A12, FLA_ONE, A22 );
   /*------------------------------------------------*/
   FLA_Cont_with_3x3_to_2x2
      ( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                               A10, A11, /**/ A12,
      /* ************* */  /* **************** */
        &ABL, /**/ &ABR,       A20, A21, /**/ A22,
        FLA_TL );
 }
```

Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} = \text{LU}(A_{11})$
$A_{12} = \text{TRILU}(A_{11})^{-1} A_{12}$
$A_{21} = A_{21} \text{TRIU}(A_{11})^{-1}$
$A_{22} = A_{22} - A_{21} A_{12}$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

```
FLA_Part_2x2( A,     &ATL, &ATR,
                     &ABL, &ABR,     0, 0, FLA_TL );

while (FLA_Obj_length( ATL ) < FLA_Obj_length( A )){

  b = min( FLA_Obj_length( ABR ), nb_alg );

  FLA_Repart_2x2_to_3x3
     ( ATL, /**/ ATR,         &A00, /**/ &A01, &A02,
    /* ************* */    /* ******************** */
                              &A10, /**/ &A11, &A12,
       ABL, /**/ ABR,         &A20, /**/ &A21, &A22,
       b, b, FLA_BR );
  /*------------------------------------------------*/
  LU_unb_var5( A11 );

  FLA_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
            FLA_NO_TRANSPOSE, FLA_UNIT_DIAG,
            FLA_ONE, A11, A12 );

  FLA_Trsm( FLA_RIGHT, FLA_UPPER_TRIANGULAR,
            FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
            FLA_ONE, A11, A21 );

  FLA_Gemm( FLA_NO_TRANSPOSE, FLA_NO_TRANSPOSE,
            FLA_MINUS_ONE, A21, A12, FLA_ONE, A22 );
  /*------------------------------------------------*/
  FLA_Cont_with_3x3_to_2x2
     ( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                              A10, A11, /**/ A12,
     /* ************** */  /* ****************** */
       &ABL, /**/ &ABR,       A20, A21, /**/ A22,
       FLA_TL );
}
```

Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} = \text{LU}(A_{11})$
$A_{12} = \text{TRILU}(A_{11})^{-1} A_{12}$
$A_{21} = A_{21} \text{TRIU}(A_{11})^{-1}$
$A_{22} = A_{22} - A_{21} A_{12}$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

```
FLA_Part_2x2( A,    &ATL, &ATR,
                     &ABL, &ABR,     0, 0, FLA_TL );

 while (FLA_Obj_length( ATL ) < FLA_Obj_length( A )){

   b = min( FLA_Obj_length( ABR ), nb_alg );

   FLA_Repart_2x2_to_3x3
      ( ATL, /**/ ATR,        &A00, /**/ &A01, &A02,
     /* ************* */   /* ******************** */
                              &A10, /**/ &A11, &A12,
        ABL, /**/ ABR,        &A20, /**/ &A21, &A22,
        b, b, FLA_BR );
   /*------------------------------------------------*/
   LU_unb_var5( A11 );

   FLA_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
             FLA_NO_TRANSPOSE, FLA_UNIT_DIAG,
             FLA_ONE, A11, A12 );

   FLA_Trsm( FLA_RIGHT, FLA_UPPER_TRIANGULAR,
             FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
             FLA_ONE, A11, A21 );

   FLA_Gemm( FLA_NO_TRANSPOSE, FLA_NO_TRANSPOSE,
             FLA_MINUS_ONE, A21, A12, FLA_ONE, A22 );
   /*------------------------------------------------*/
   FLA_Cont_with_3x3_to_2x2
      ( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                               A10, A11, /**/ A12,
     /* ************** */   /* ****************** */
        &ABL, /**/ &ABR,       A20, A21, /**/ A22,
        FLA_TL );
 }
```

Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array} \right)$$

where A_{11} is $b \times b$

$A_{11} = \text{LU}(A_{11})$

$A_{12} = \text{TRILU}(A_{11})^{-1} A_{12}$

$A_{21} = A_{21} \text{TRIU}(A_{11})^{-1}$

$A_{22} = A_{22} - A_{21} A_{12}$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array} \right)$$

endwhile

```
FLA_Part_2x2( A,    &ATL, &ATR,
                     &ABL, &ABR,     0, 0, FLA_TL );

 while (FLA_Obj_length( ATL ) < FLA_Obj_length( A )){

   b = min( FLA_Obj_length( ABR ), nb_alg );

   FLA_Repart_2x2_to_3x3
      ( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
    /* ************* */   /* ******************** */
                             &A10, /**/ &A11, &A12,
        ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
        b, b, FLA_BR );
   /*------------------------------------------------*/
   LU_unb_var5( A11 );

   FLA_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
             FLA_NO_TRANSPOSE, FLA_UNIT_DIAG,
             FLA_ONE, A11, A12 );

   FLA_Trsm( FLA_RIGHT, FLA_UPPER_TRIANGULAR,
             FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
             FLA_ONE, A11, A21 );

   FLA_Gemm( FLA_NO_TRANSPOSE, FLA_NO_TRANSPOSE,
             FLA_MINUS_ONE, A21, A12, FLA_ONE, A22 );
   /*------------------------------------------------*/
   FLA_Cont_with_3x3_to_2x2
      ( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                               A10, A11, /**/ A12,
     /* ************** */   /* ****************** */
        &ABL, /**/ &ABR,       A20, A21, /**/ A22,
        FLA_TL );
 }
```

Algorithm: $[A] := \text{LU\_BLK\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

  where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

  Determine block size b

  Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

  where A_{11} is $b \times b$

$$\boxed{\begin{array}{l} A_{11} = \text{LU}(A_{11}) \\ A_{12} = \text{TRILU}(A_{11})^{-1} A_{12} \\ A_{21} = A_{21} \text{TRIU}(A_{11})^{-1} \\ A_{22} = A_{22} - A_{21} A_{12} \end{array}}$$

  Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

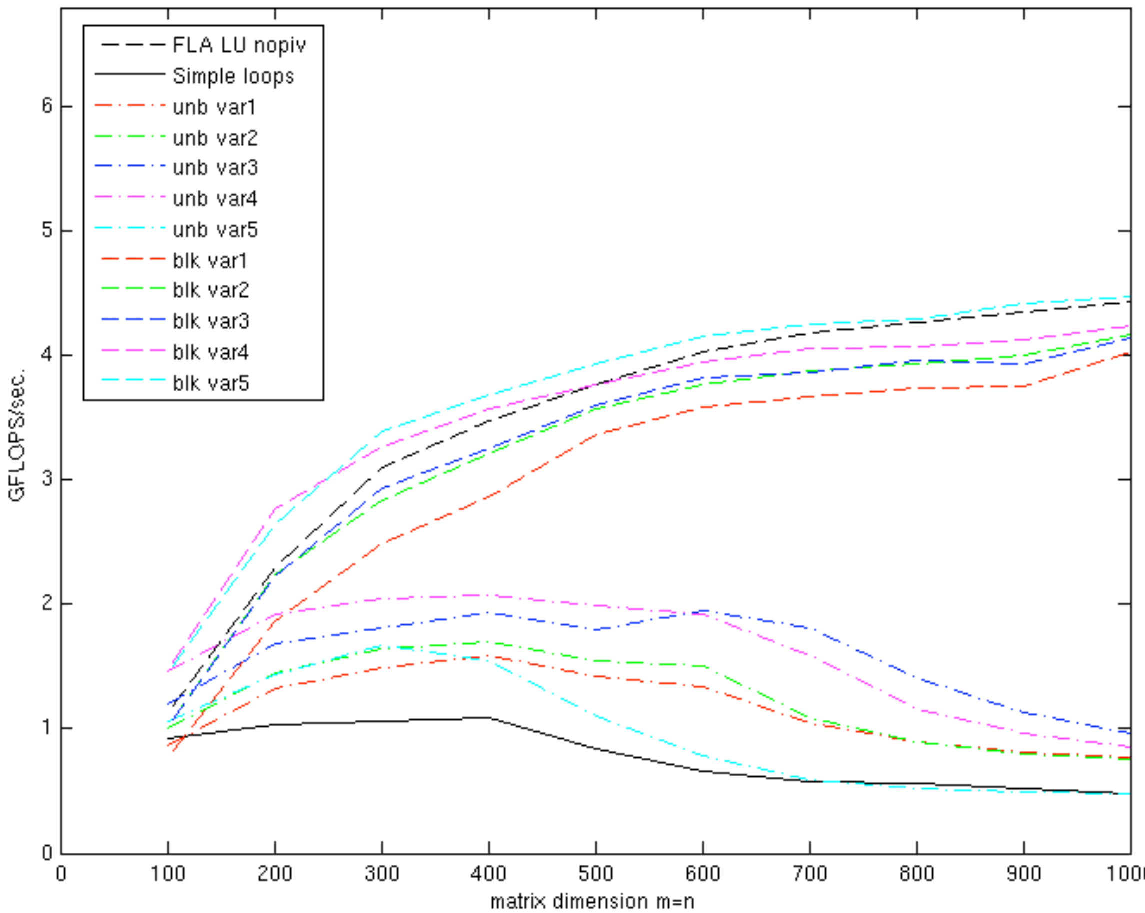
```
FLA_Part_2x2( A,    &ATL, &ATR,
                     &ABL, &ABR,     0, 0, FLA_TL );

 while (FLA_Obj_length( ATL ) < FLA_Obj_length( A )){

   b = min( FLA_Obj_length( ABR ), nb_alg );

   FLA_Repart_2x2_to_3x3
      ( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
     /* ************* */   /* ****************** */
                             &A10, /**/ &A11, &A12,
        ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
        b, b, FLA_BR );
   /*------------------------------------------------------------*/
   LU_unb_var5( A11 );

   FLA_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
             FLA_NO_TRANSPOSE, FLA_UNIT_DIAG,
             FLA_ONE, A11, A12 );

   FLA_Trsm( FLA_RIGHT, FLA_UPPER_TRIANGULAR,
             FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
             FLA_ONE, A11, A21 );

   FLA_Gemm( FLA_NO_TRANSPOSE, FLA_NO_TRANSPOSE,
             FLA_MINUS_ONE, A21, A12, FLA_ONE, A22 );
   /*------------------------------------------------------------*/
   FLA_Cont_with_3x3_to_2x2
      ( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                               A10, A11, /**/ A12,
      /* ************** */  /* ***************** */
        &ABL, /**/ &ABR,       A20, A21, /**/ A22,
        FLA_TL );
 }
```

An algorithm for every occasion

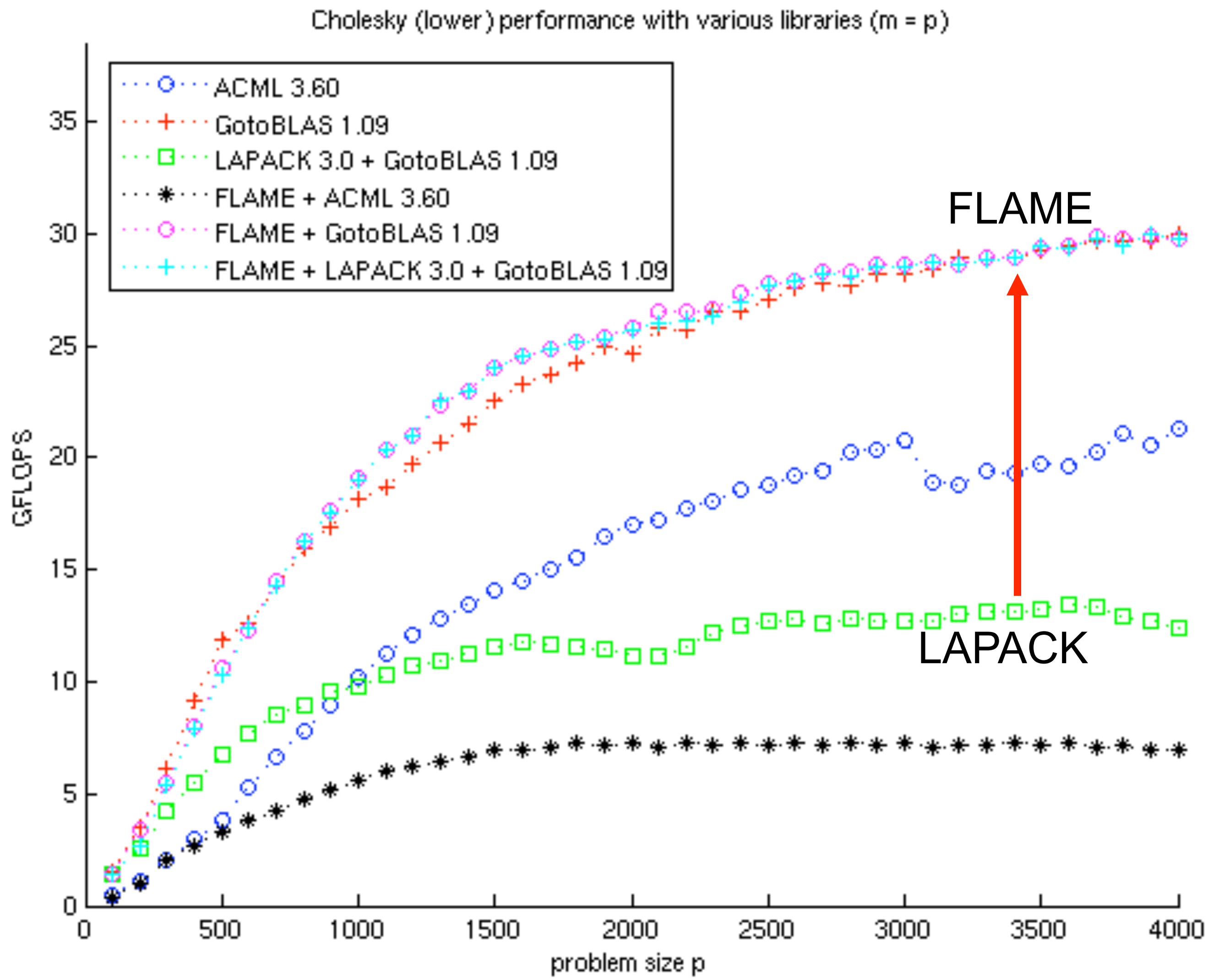
- There are five loop-based algorithms for LU factorization without pivoting
  - Five unblocked
  - Five blocked

- On different architectures different algorithm will achieve best performance

Performance (Intel Xeon 3.4GHz)



AMD Opteron 8 cores (2.4GHz)



Algorithms-by-Blocks

(some call these tiled algorithms)

- Distinguish
  - Blocked algorithms
  - Blocked algorithms that copy into contiguous blocks as an intermediate step
  - Algorithms-by-blocks (tiled algorithms)
- Fred Gustavson indicates he anticipated the need for algorithms-by-blocks in 1986.
- First published use of storage by block: TR by Greg Henry, around 1992
- Many papers by Fred Gustavson, Bo Kagstrom and colleagues during late 1990s and beyond
- Many efforts to hide nastiness:
  - Skjellum et al, Wise, …
- Application to OOC: SOLAR library by Toledo and Gustavson and POOCLAPACK from UT-Austin
- Our first effort: The FLASH extension of the FLAME/C API (FLAWN#12, 2004)
- Very recently: PLASMA project at UTK and SuperMatrix project at UT-Austin (2006-07)

The fundamental problem

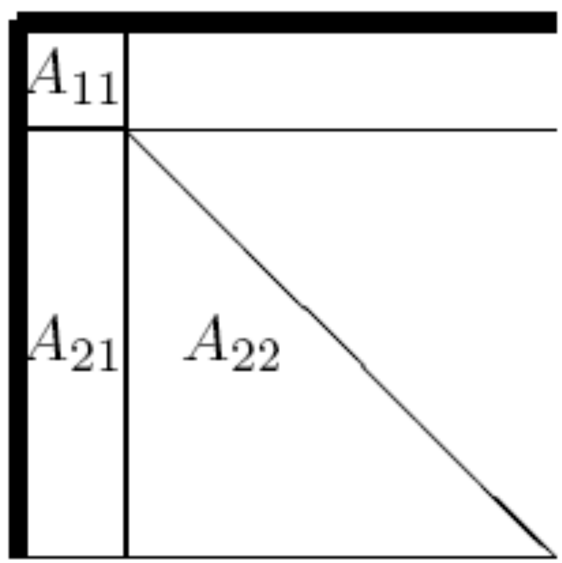
- People like Kazushige Goto optimize so well that it has been hard to show the benefits of storage by blocks.

- This changes with the emergence of multicore architectures(?).

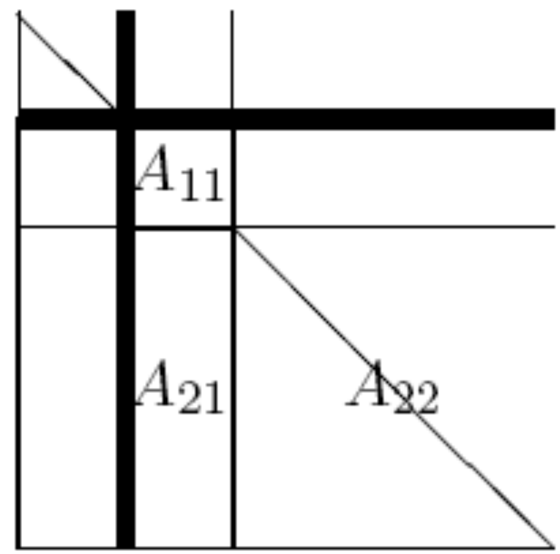
Overview

- A bit of history
- Programming algorithms-by-blocks is ~~hard~~ easy
  - An example: Cholesky factorization by blocks
- Programming algorithms-by-blocks ~~further~~ simplifies ~~complicates~~ programming (future) multicore architectures
- Conclusion
- Resources

Example: Cholesky factorization



Iteration 1



Iteration 2

Algorithm: $A := \textsc{Chol\_blk\_var3}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

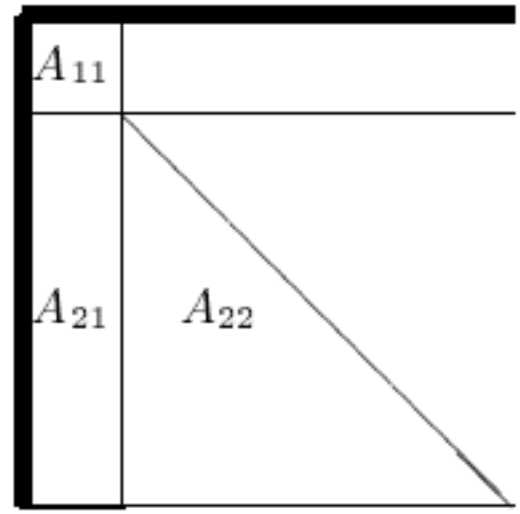
where A_{11} is $b \times b$

$$A_{11} = \mathrm{Chol}(A_{11})$$
$$A_{21} = A_{21}\,\mathrm{TRIL}\,(A_{11})^{-T}$$
$$A_{22} = A_{22} - \mathrm{TRIL}\,(A_{21}A_{21}^T)$$

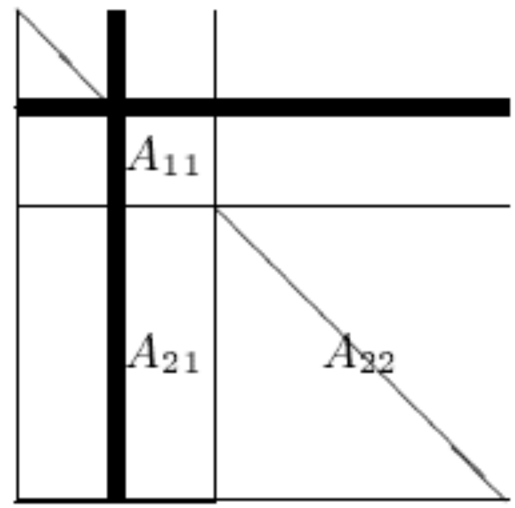
Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

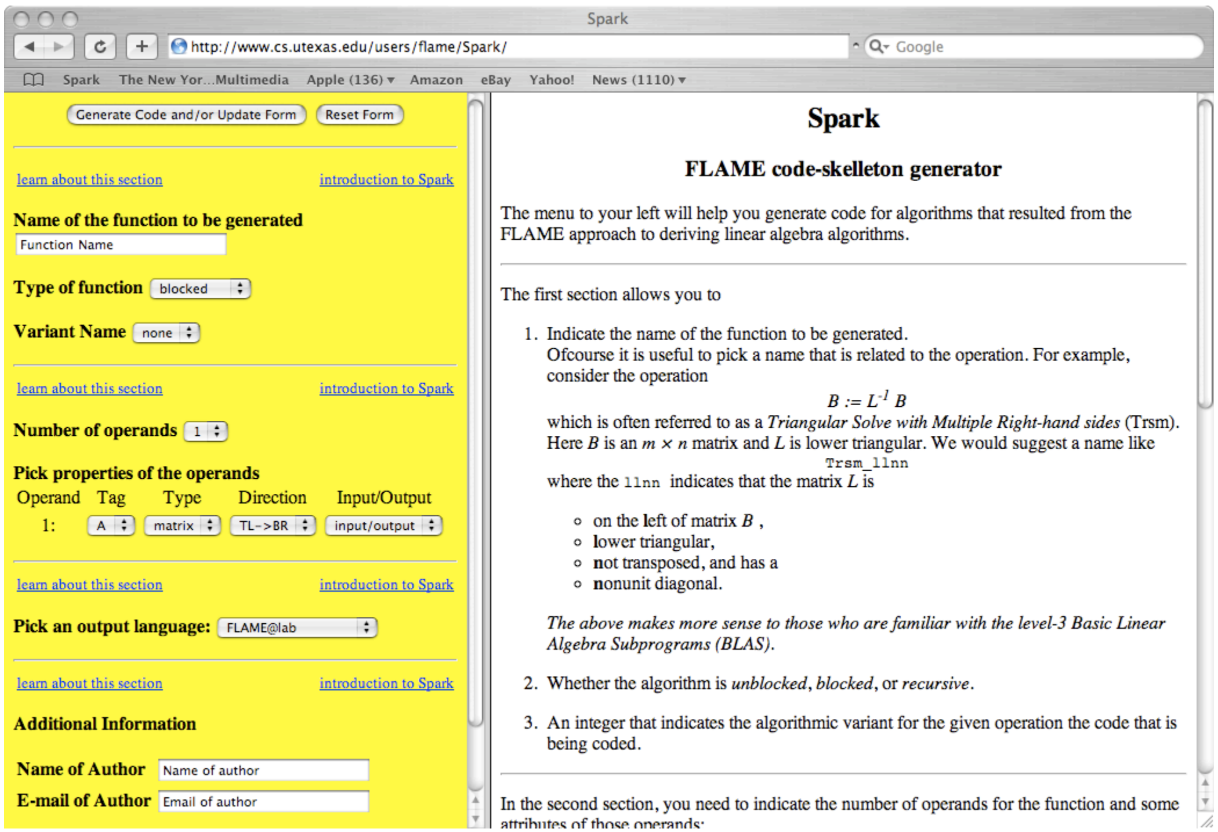


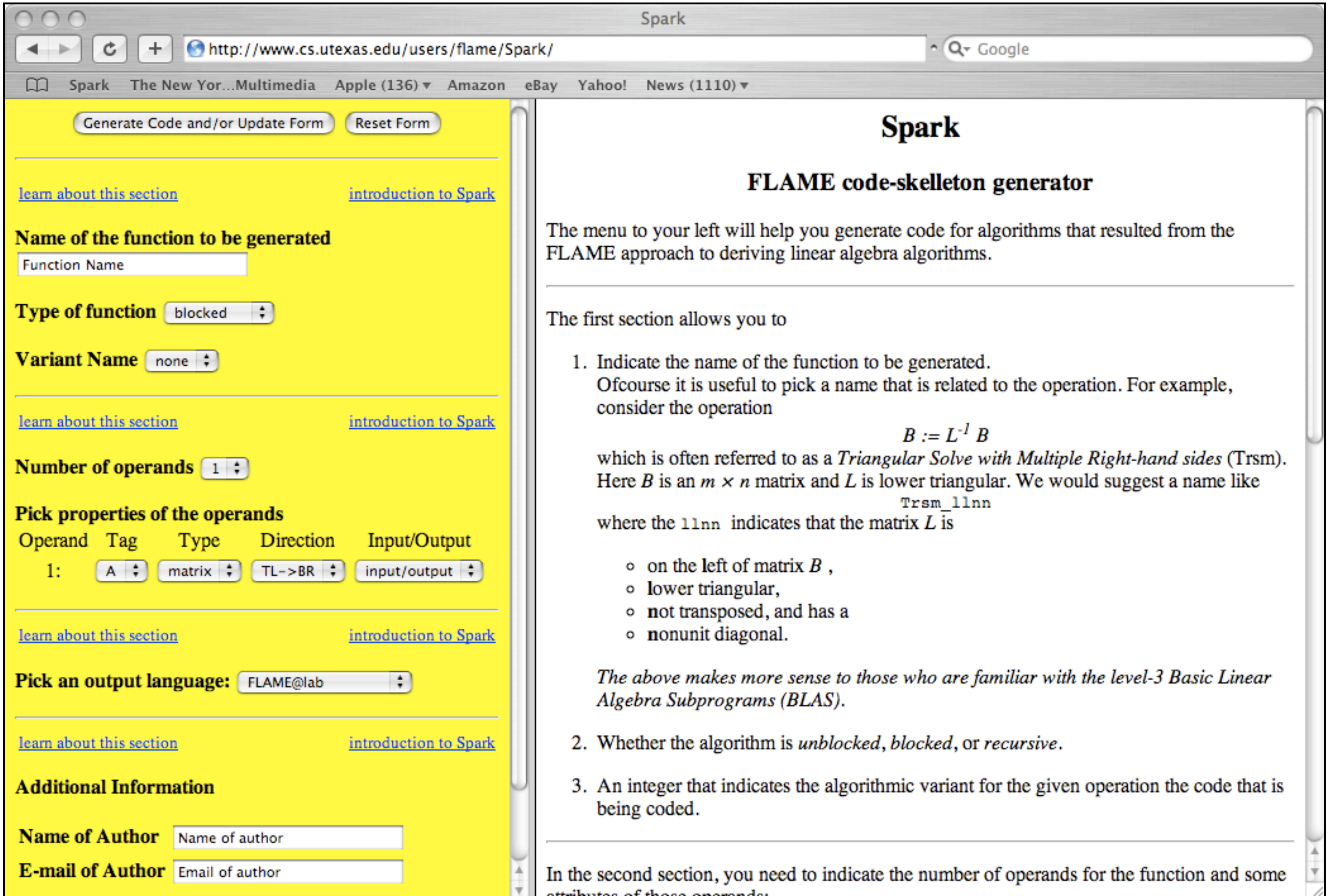
Iteration 1

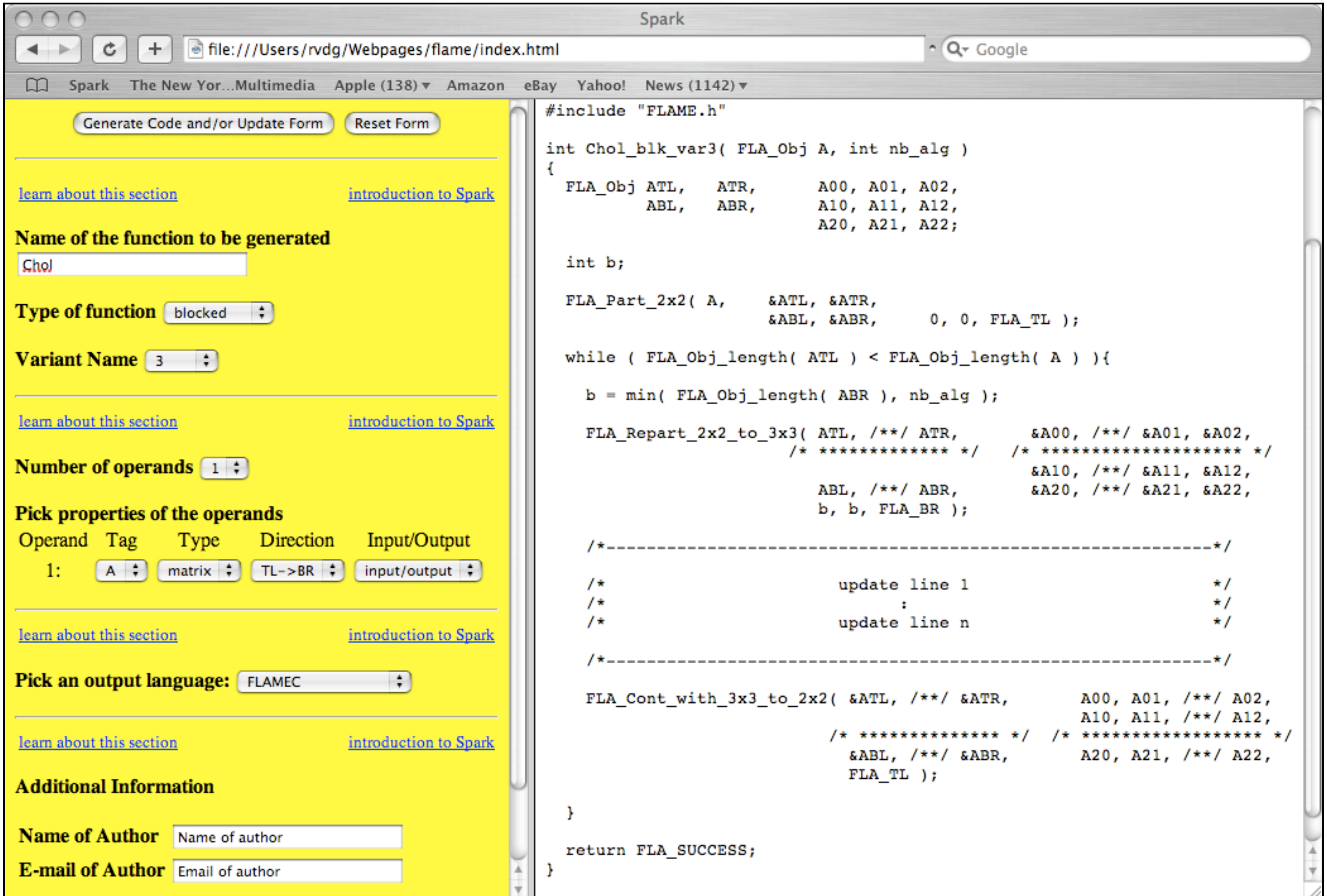


Iteration 2

Coding the Algorithm-by-Blocks in 15 minutes




Spark
http://www.cs.utexas.edu/users/flame/Spark/
Google
Spark The New Yor...Multimedia Apple (136) Amazon eBay Yahoo! News (1110)
Generate Code and/or Update Form
Reset Form
learn about this section
introduction to Spark
Name of the function to be generated
Function Name
Type of function
blocked
Variant Name
none
learn about this section
introduction to Spark
Number of operands
1
Pick properties of the operands
Operand Tag Type Direction Input/Output
1: A matrix TL->BR input/output
learn about this section
introduction to Spark
Pick an output language:
FLAME@lab
learn about this section
introduction to Spark
Additional Information
Name of Author
Name of author
E-mail of Author
Email of author
Spark
FLAME code-skelleton generator
The menu to your left will help you generate code for algorithms that resulted from the FLAME approach to deriving linear algebra algorithms.
The first section allows you to
1. Indicate the name of the function to be generated.
Ofcourse it is useful to pick a name that is related to the operation. For example, consider the operation
$B := L^{-1} B$
which is often referred to as a *Triangular Solve with Multiple Right-hand sides* (Trsm). Here B is an $m \times n$ matrix and L is lower triangular. We would suggest a name like
Trsm_llnn
where the llnn indicates that the matrix L is
on the left of matrix B ,
lower triangular,
not transposed, and has a
nonunit diagonal.
The above makes more sense to those who are familiar with the level-3 Basic Linear Algebra Subprograms (BLAS).
2. Whether the algorithm is *unblocked*, *blocked*, or *recursive*.
3. An integer that indicates the algorithmic variant for the given operation the code that is being coded.
In the second section, you need to indicate the number of operands for the function and some attributes of those operands:


Spark
file:///Users/rvdg/Webpages/flame/index.html
Google
Spark
The New Yor...Multimedia
Apple (138)
Amazon
eBay
Yahoo!
News (1142)
Generate Code and/or Update Form
Reset Form
learn about this section
introduction to Spark
Name of the function to be generated
Chol
Type of function
blocked
Variant Name
3
learn about this section
introduction to Spark
Number of operands
1
Pick properties of the operands
Operand
Tag
Type
Direction
Input/Output
1:
A
matrix
TL->BR
input/output
learn about this section
introduction to Spark
Pick an output language:
FLAMEC
learn about this section
introduction to Spark
Additional Information
Name of Author
Name of author
E-mail of Author
Email of author
#include "FLAME.h"
int Chol_blk_var3(FLA_Obj A, int nb_alg)
{
FLA_Obj ATL, ATR, A00, A01, A02,
ABL, ABR, A10, A11, A12,
A20, A21, A22;
int b;
FLA_Part_2x2(A, &ATL, &ATR,
&ABL, &ABR, 0, 0, FLA_TL);
while (FLA_Obj_length(ATL) < FLA_Obj_length(A)){
b = min(FLA_Obj_length(ABR), nb_alg);
FLA_Repart_2x2_to_3x3(ATL, /**/ ATR, &A00, /**/ &A01, &A02,
/* ************* */ /* ******************** */
&A10, /**/ &A11, &A12,
ABL, /**/ ABR, &A20, /**/ &A21, &A22,
b, b, FLA_BR);
/*--*/
/* update line 1 */
/* : */
/* update line n */
/*--*/
FLA_Cont_with_3x3_to_2x2(&ATL, /**/ &ATR, A00, A01, /**/ A02,
A10, A11, /**/ A12,
/* ************** */ /* ****************** */
&ABL, /**/ &ABR, A20, A21, /**/ A22,
FLA_TL);
}
return FLA_SUCCESS;
}

Chol_blk_var3.c

```c
int Chol_blk_var3( FLA_Obj A, int nb_alg )
{
  FLA_Obj ATL,   ATR,      A00, A01, A02,
          ABL,   ABR,      A10, A11, A12,
                           A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           b, b, FLA_BR );

    /*------------------------------------------------------------*/

    /*                       update line 1                         */
    /*                             :                               */
    /*                       update line n                         */

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );

  }
```

1:** Chol_blk_var3.c   2% (36,0)   (C/l Abbrev)

Algorithm: $A := \text{CHOL\_BLK\_VAR3}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} = \text{Chol}(A_{11})$

$A_{21} = A_{21}\,\text{TRIL}\,(A_{11})^{-T}$

$A_{22} = A_{22} - \text{TRIL}\,(A_{21}A_{21}^T)$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

Chol_blk_var3.c

```c
int Chol_blk_var3( FLA_Obj A, int nb_alg )
{
  FLA_Obj ATL,   ATR,      A00, A01, A02,
          ABL,   ABR,      A10, A11, A12,
                           A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           b, b, FLA_BR );

    /*------------------------------------------------------------*/
    FLA_Chol_unb_var3( A11 );
    FLA_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
              FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, A11, A21 );
    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, FLA_MINUS_ONE,
              A21, FLA_ONE, A22 );
    /*------------------------------------------------------------*/

    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
```

1:** Chol_blk_var3.c 2% (31,68) (C/l Abbrev)

Algorithm: $A := \textsc{Chol\_blk\_var3}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} = \mathrm{Chol}(A_{11})$

$A_{21} = A_{21}\,\mathrm{TRIL}\,(A_{11})^{-T}$

$A_{22} = A_{22} - \mathrm{TRIL}\,(A_{21}A_{21}^T)$

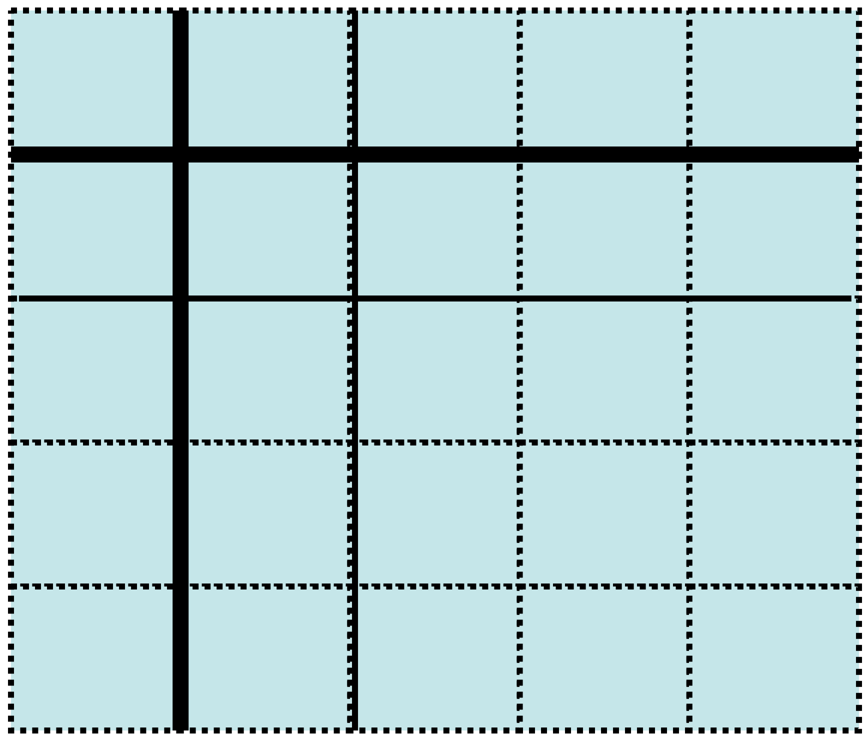
Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

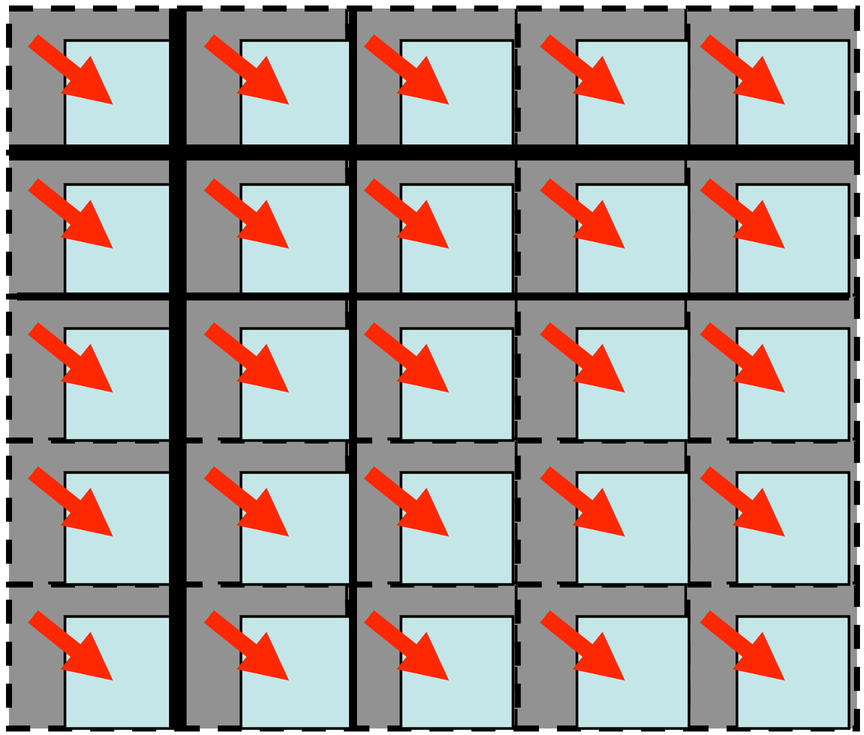
endwhile

Blocked Algorithm

Algorithm-by-Blocks

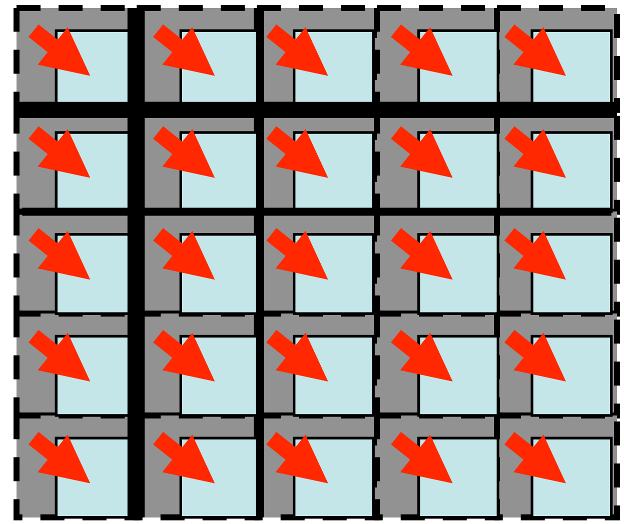


Storage-by-Blocks



Chol_blk_var3.c

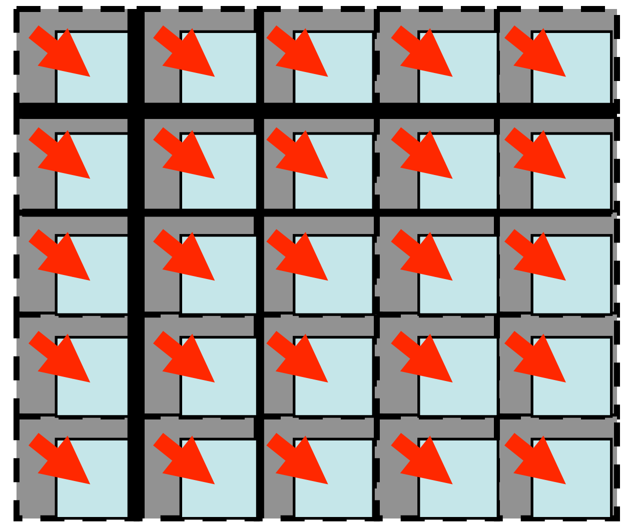
```c
int Chol_blk_var3( FLA_Obj A, int nb_alg )
{
  FLA_Obj ATL,   ATR,      A00, A01, A02,
          ABL,   ABR,      A10, A11, A12,
                           A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           b, b, FLA_BR );

    /*------------------------------------------------------------*/
    FLA_Chol_unb_var3( A11 );
    FLA_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
              FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, A11, A21 );
    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, FLA_MINUS_ONE,
              A21, FLA_ONE, A22 );
    /*------------------------------------------------------------*/
    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
```



1:** Chol_blk_var3.c   2% (31,68)   (C/l Abbrev)

Chol_blk_var3.c

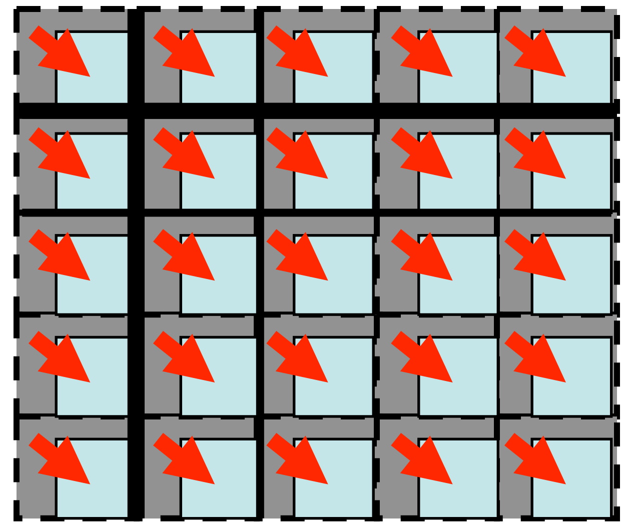
```
int Chol_blk_var3( FLA_Obj A, int nb_alg )
{
  FLA_Obj ATL,   ATR,      A00, A01, A02,
          ABL,   ABR,      A10, A11, A12,
                           A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           b, b, FLA_BR );

    /*------------------------------------------------------------*/
    FLA_Chol_unb_var3( A11 );
    FLA_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
              FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, A11, A21 );
    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, FLA_MINUS_ONE,
              A21, FLA_ONE, A22 );
    /*------------------------------------------------------------*/
    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */   /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
```



1:** Chol_blk_var3.c   2% (3,28)   (C/l Abbrev)

Chol_blk_var3.c

```c
int Chol_blk_var3( FLA_Obj A )
{
  FLA_Obj ATL,    ATR,       A00, A01, A02,
          ABL,    ABR,       A10, A11, A12,
                             A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           b, b, FLA_BR );

    /*------------------------------------------------------------*/
    FLA_Chol_unb_var3( A11 );
    FLA_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
              FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, A11, A21 );
    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, FLA_MINUS_ONE,
              A21, FLA_ONE, A22 );
    /*------------------------------------------------------------*/

    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
```

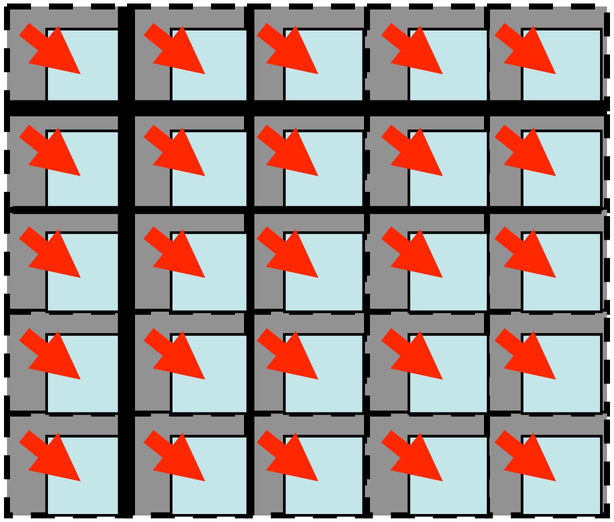


1:** Chol_blk_var3.c 2% (22,32) (C/l Abbrev)

Chol_blk_var3.c

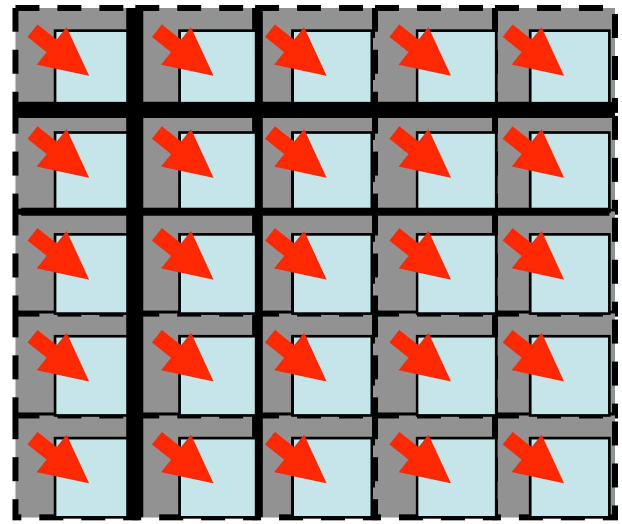
```
int Chol_blk_var3( FLA_Obj A )
{
  FLA_Obj ATL,    ATR,       A00, A01, A02,
          ABL,    ABR,       A10, A11, A12,
                             A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           1, 1, FLA_BR );

    /*------------------------------------------------------------*/
    FLA_Chol_unb_var3( FLASH_MATRIX_AT( A11 ) );
    FLA_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
              FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, A11, A21 );
    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, FLA_MINUS_ONE,
              A21, FLA_ONE, A22 );
    /*------------------------------------------------------------*/

    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
```

1:** Chol_blk_var3.c   2% (25,45)   (C/l Abbrev)



Chol_blk_var3.c

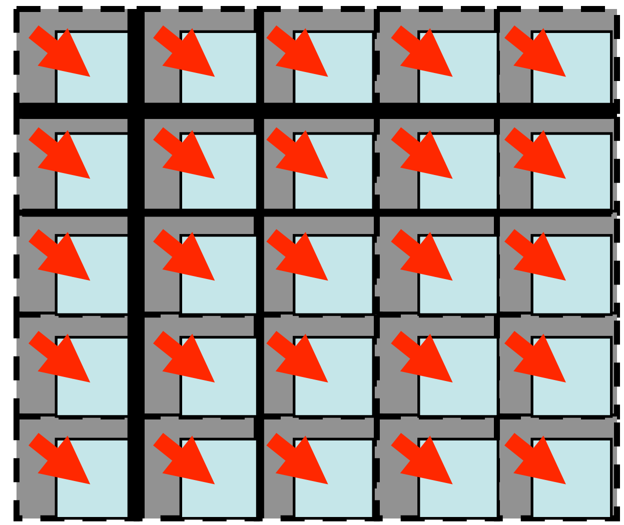
```
int Chol_blk_var3( FLA_Obj A )
{
  FLA_Obj ATL,   ATR,      A00, A01, A02,
          ABL,   ABR,      A10, A11, A12,
                           A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           1, 1, FLA_BR );

    /*------------------------------------------------------------*/
    FLA_Chol_unb_var3( FLASH_MATRIX_AT( A11 ) );
    FLASH_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
                FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
                FLA_ONE, A11, A21 );
    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, FLA_MINUS_ONE,
                A21, FLA_ONE, A22 );
    /*------------------------------------------------------------*/
    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
```



1:** Chol_blk_var3.c   2% (26,9)   (C/l Abbrev)

Chol_blk_var3.c

```c
int Chol_blk_var3( FLA_Obj A )
{
  FLA_Obj ATL,   ATR,      A00, A01, A02,
          ABL,   ABR,      A10, A11, A12,
                           A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                         /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           1, 1, FLA_BR );

    /*------------------------------------------------------------*/
    FLA_Chol_unb_var3( FLASH_MATRIX_AT( A11 ) );
    FLASH_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
                FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
                FLA_ONE, A11, A21 );
    FLASH_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, FLA_MINUS_ONE,
                A21, FLA_ONE, A22 );
    /*------------------------------------------------------------*/
    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */   /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
```

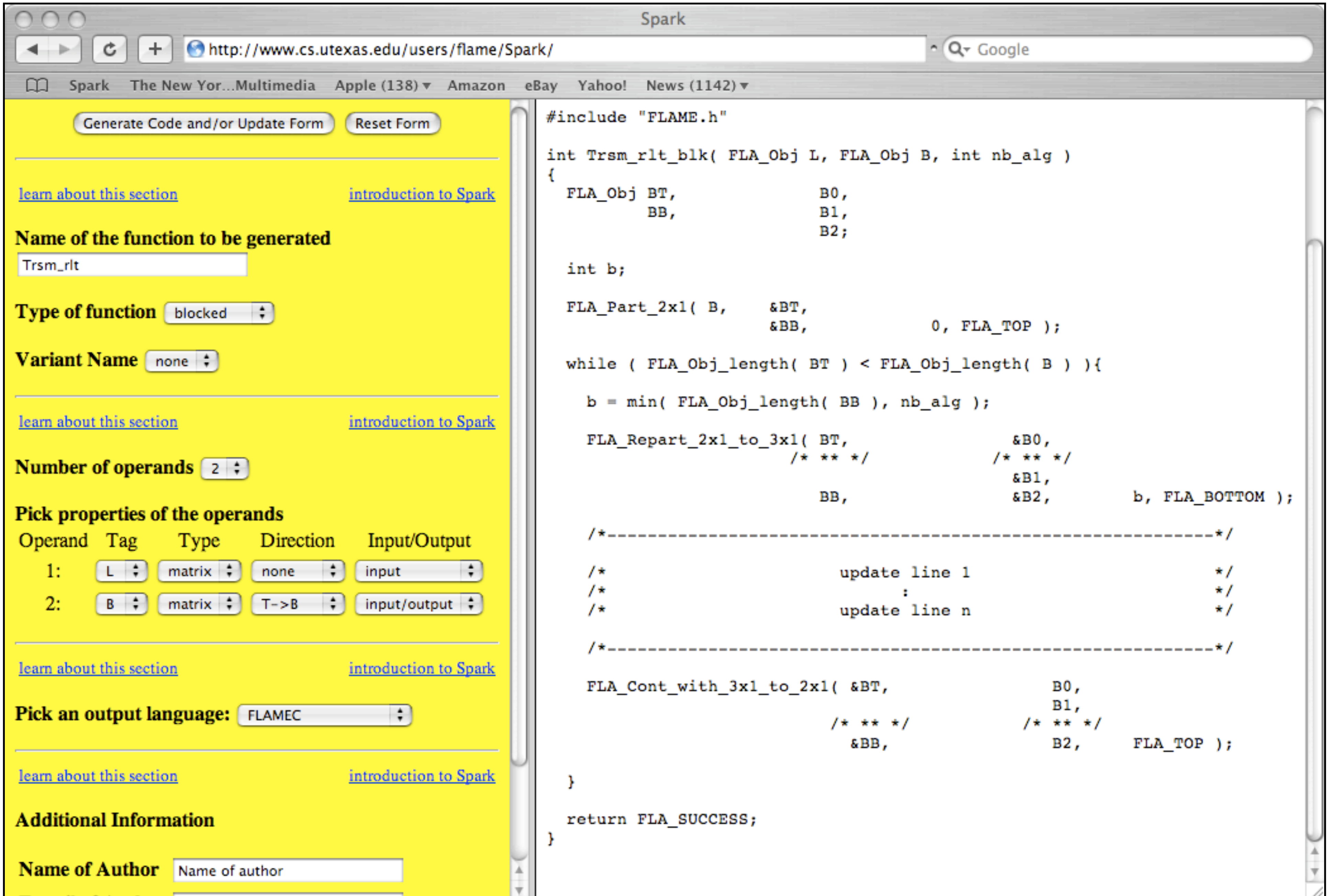


1:** Chol_blk_var3.c   2% (29,9)   (C/l Abbrev)

Chol_blk_var3.c

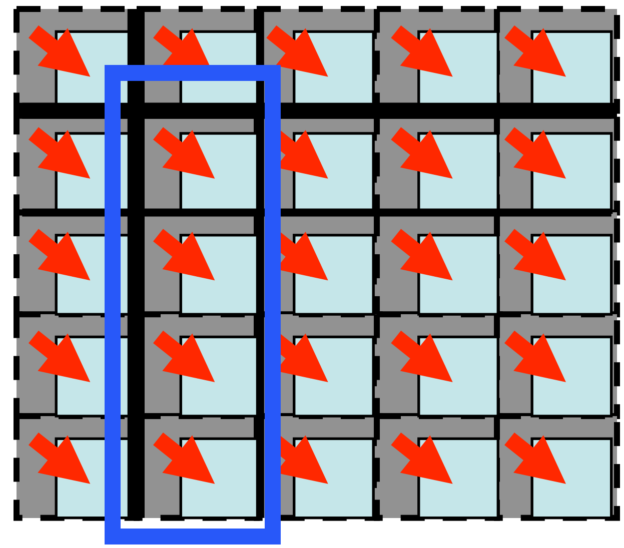
```
int Chol_blk_var3( FLA_Obj A )
{
  FLA_Obj ATL,   ATR,      A00, A01, A02,
          ABL,   ABR,      A10, A11, A12,
                           A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           1, 1, FLA_BR );

    /*------------------------------------------------------------*/
    FLA_Chol_unb_var3( FLASH_MATRIX_AT( A11 ) );
    FLASH_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
                FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
                FLA_ONE, A11, A21 );
    FLASH_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, FLA_MINUS_ONE,
                A21, FLA_ONE, A22 );
    /*------------------------------------------------------------*/
    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
1:**  Chol_blk_var3.c     2% (27,16)     (C/l Abbrev)
```

But doesn't this just mean that the FLASH_Trsm and FLASH_Syrk routines are hard to code?


Spark
http://www.cs.utexas.edu/users/flame/Spark/
Google
Spark The New Yor...Multimedia Apple (138) Amazon eBay Yahoo! News (1142)
Generate Code and/or Update Form
Reset Form
learn about this section
introduction to Spark
Name of the function to be generated
Trsm_rlt
Type of function
blocked
Variant Name
none
learn about this section
introduction to Spark
Number of operands
2
Pick properties of the operands
Operand Tag Type Direction Input/Output
1: L matrix none input
2: B matrix T->B input/output
learn about this section
introduction to Spark
Pick an output language:
FLAMEC
learn about this section
introduction to Spark
Additional Information
Name of Author
Name of author
#include "FLAME.h"
int Trsm_rlt_blk(FLA_Obj L, FLA_Obj B, int nb_alg)
{
FLA_Obj BT, B0,
BB, B1,
B2;
int b;
FLA_Part_2x1(B, &BT,
&BB, 0, FLA_TOP);
while (FLA_Obj_length(BT) < FLA_Obj_length(B)){
b = min(FLA_Obj_length(BB), nb_alg);
FLA_Repart_2x1_to_3x1(BT, &B0,
/* ** */ /* ** */
&B1,
BB, &B2, b, FLA_BOTTOM);
/*--*/
/* update line 1 */
/* : */
/* update line n */
/*--*/
FLA_Cont_with_3x1_to_2x1(&BT, B0,
B1,
/* ** */ /* ** */
&BB, B2, FLA_TOP);
}
return FLA_SUCCESS;
}

Chol_blk_var3.c

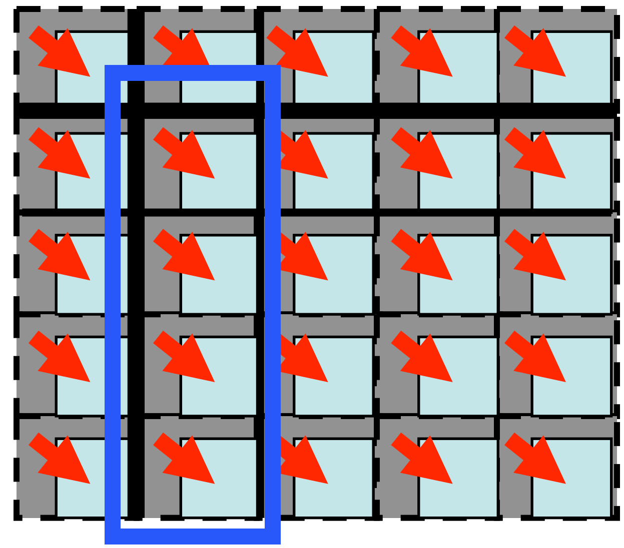
```
int Trsm_rlt_blk( FLA_Obj L, FLA_Obj B, int nb_alg )
{
  FLA_Obj BT,              B0,
          BB,              B1,
                           B2;

  int b;

  FLA_Part_2x1( B,    &BT,
                      &BB,            0, FLA_TOP );

  while ( FLA_Obj_length( BT ) < FLA_Obj_length( B ) ){

    b = min( FLA_Obj_length( BB ), nb_alg );

    FLA_Repart_2x1_to_3x1( BT,                &B0,
                        /* ** */            /* ** */
                                              &B1,
                           BB,                &B2,        b, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    /*                         update line 1                      */
    /*                               :                            */
    /*                         update line n                      */

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x1_to_2x1( &BT,                B0,
                                                  B1,
                            /* ** */            /* ** */
                              &BB,                B2,     FLA_TOP );

  }
```



1:** Chol_blk_var3.c 2% (3,0) (C/l Abbrev)

Chol_blk_var3.c

```
int Trsm_rlt_blk( FLA_Obj L, FLA_Obj B, int nb_alg )
{
  FLA_Obj BT,              B0,
          BB,              B1,
                           B2;

  int b;

  FLA_Part_2x1( B,    &BT,
                      &BB,            0, FLA_TOP );

  while ( FLA_Obj_length( BT ) < FLA_Obj_length( B ) ){

    b = min( FLA_Obj_length( BB ), nb_alg );

    FLA_Repart_2x1_to_3x1( BT,                &B0,
                        /* ** */           /* ** */
                                              &B1,
                           BB,                &B2,        b, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    /*                        update line 1                       */
    /*                              :                             */
    /*                        update line n                       */

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x1_to_2x1( &BT,                B0,
                                                  B1,
                            /* ** */          /* ** */
                              &BB,                B2,     FLA_TOP );

  }
```

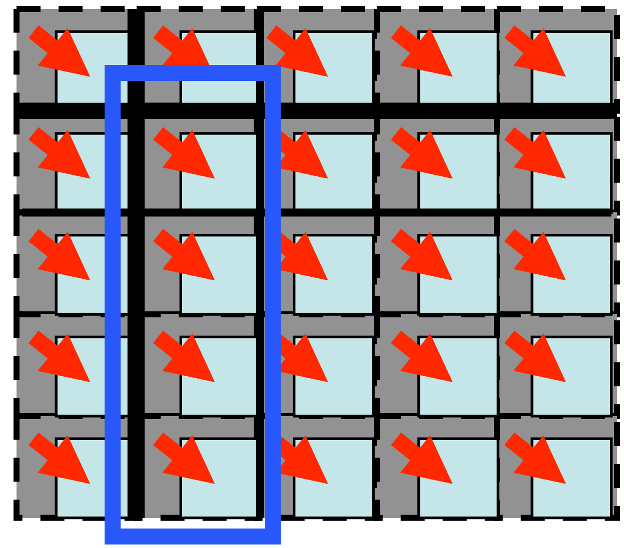


1:** Chol_blk_var3.c 2% (3,50) (C/l Abbrev)

Chol_blk_var3.c

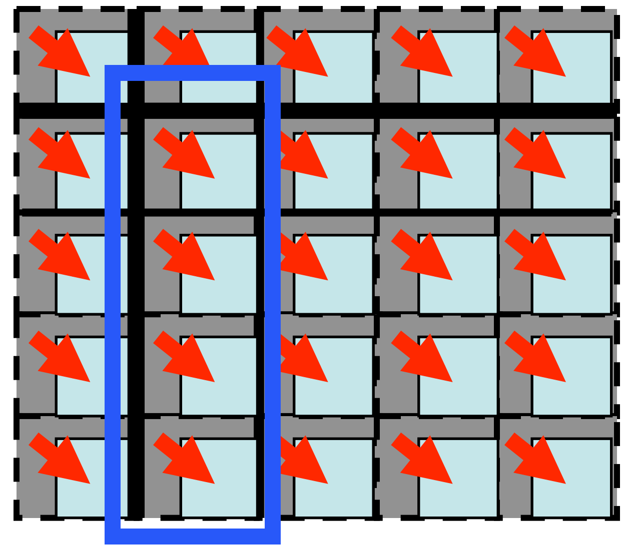
```
int Trsm_rlt_blk( FLA_Obj L, FLA_Obj B )
{
  FLA_Obj BT,              B0,
          BB,              B1,
                           B2;

  int b;

  FLA_Part_2x1( B,    &BT,
                      &BB,            0, FLA_TOP );

  while ( FLA_Obj_length( BT ) < FLA_Obj_length( B ) ){

    b = min( FLA_Obj_length( BB ), nb_alg );

    FLA_Repart_2x1_to_3x1( BT,                &B0,
                        /* ** */            /* ** */
                                              &B1,
                           BB,                &B2,        b, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    /*                      update line 1                         */
    /*                            :                               */
    /*                      update line n                         */

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x1_to_2x1( &BT,                B0,
                                                  B1,
                            /* ** */            /* ** */
                              &BB,                B2,     FLA_TOP );

  }
```

1:** Chol_blk_var3.c 2% (16,0) (C/l Abbrev)



Chol_blk_var3.c

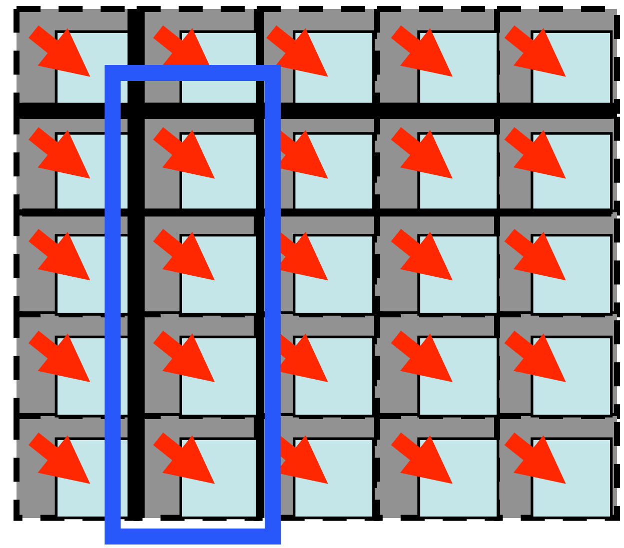
```
int Trsm_rlt_blk( FLA_Obj L, FLA_Obj B )
{
  FLA_Obj BT,              B0,
          BB,              B1,
                           B2;

  int b;

  FLA_Part_2x1( B,    &BT,
                      &BB,            0, FLA_TOP );

  while ( FLA_Obj_length( BT ) < FLA_Obj_length( B ) ){



    FLA_Repart_2x1_to_3x1( BT,                &B0,
                        /* ** */            /* ** */
                                              &B1,
                           BB,                &B2,        b, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    /*                        update line 1                       */
    /*                              :                             */
    /*                        update line n                       */

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x1_to_2x1( &BT,                B0,
                                                  B1,
                            /* ** */            /* ** */
                              &BB,                B2,     FLA_TOP );

  }
```



1:** Chol_blk_var3.c   2% (28,0)   (C/l Abbrev)

Chol_blk_var3.c

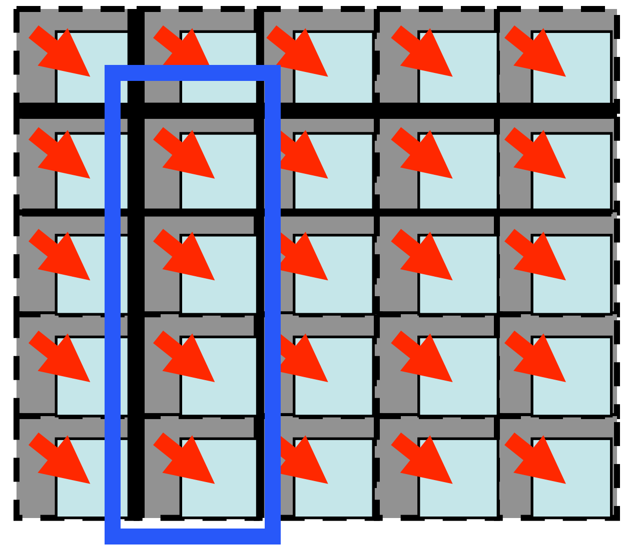
```
int Trsm_rlt_blk( FLA_Obj L, FLA_Obj B )
{
  FLA_Obj BT,              B0,
          BB,              B1,
                           B2;

  int b;

  FLA_Part_2x1( B,    &BT,
                      &BB,            0, FLA_TOP );

  while ( FLA_Obj_length( BT ) < FLA_Obj_length( B ) ){



    FLA_Repart_2x1_to_3x1( BT,                &B0,
                        /* ** */            /* ** */
                                              &B1,
                           BB,                &B2,        b, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    FLA_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
              FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, FLASH_MATRIX_AT( L ), FLASH_MATRIX_AT( B ) );

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x1_to_2x1( &BT,                B0,
                                                  B1,
                            /* ** */            /* ** */
                              &BB,                B2,     FLA_TOP );

  }
```



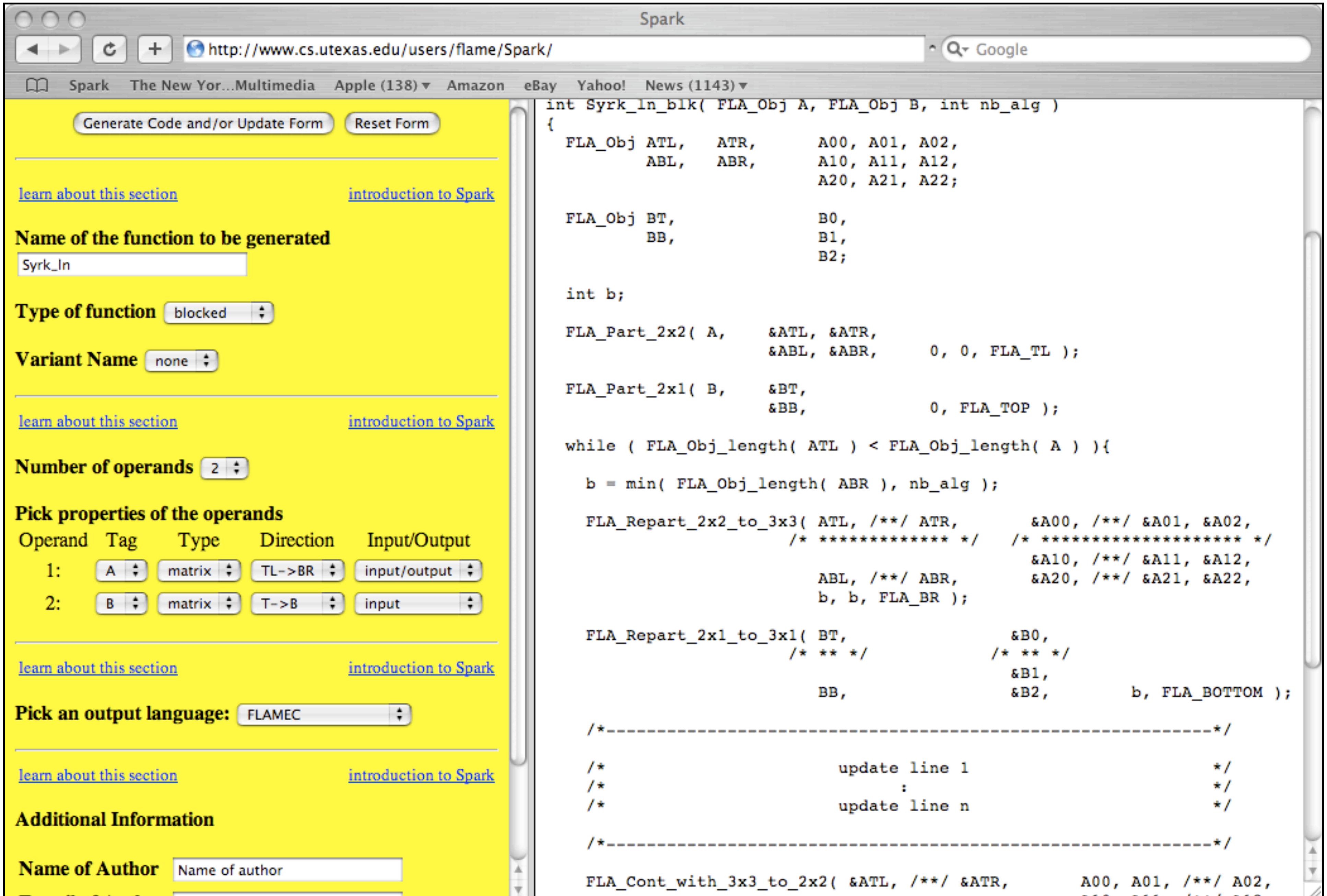
1:** Chol_blk_var3.c   2% (28,0)   (C/l Abbrev)

Chol_blk_var3.c

```
int Trsm_rlt_blk( FLA_Obj L, FLA_Obj B )
{
  FLA_Obj BT,              B0,
          BB,              B1,
                           B2;

  int b;

  FLA_Part_2x1( B,    &BT,
                      &BB,          0, FLA_TOP );

  while ( FLA_Obj_length( BT ) < FLA_Obj_length( B ) ){



    FLA_Repart_2x1_to_3x1( BT,              &B0,
                        /* ** */          /* ** */
                                          &B1,
                           BB,            &B2,        1, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    FLA_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
              FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, FLASH_MATRIX_AT( L ), FLASH_MATRIX_AT( B ) );

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x1_to_2x1( &BT,             B0,
                                               B1,
                          /* ** */          /* ** */
                              &BB,             B2,     FLA_TOP );

  }
```



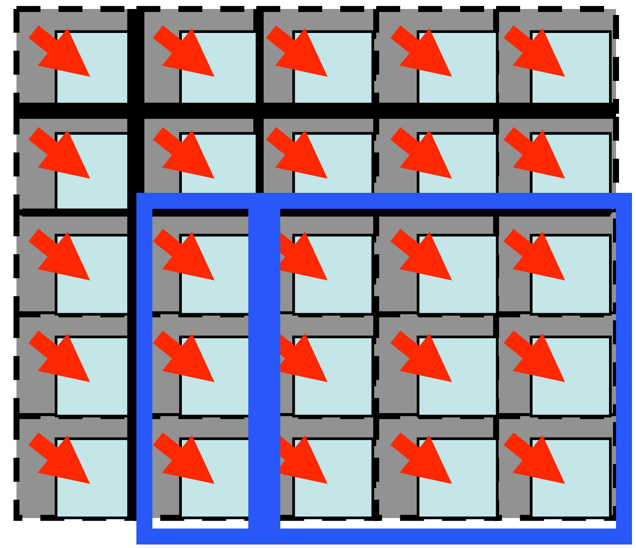
1:** Chol_blk_var3.c   2% (21,59)   (C/l Abbrev)



Spark
http://www.cs.utexas.edu/users/flame/Spark/
Google
Spark The New Yor...Multimedia Apple (138) Amazon eBay Yahoo! News (1143)
Generate Code and/or Update Form
Reset Form
learn about this section
introduction to Spark
Name of the function to be generated
Syrk_ln
Type of function
blocked
Variant Name
none
learn about this section
introduction to Spark
Number of operands
2
Pick properties of the operands
Operand Tag Type Direction Input/Output
1: A matrix TL->BR input/output
2: B matrix T->B input
learn about this section
introduction to Spark
Pick an output language:
FLAMEC
learn about this section
introduction to Spark
Additional Information
Name of Author
Name of author
int Syrk_ln_blk(FLA_Obj A, FLA_Obj B, int nb_alg)
{
FLA_Obj ATL, ATR, A00, A01, A02,
ABL, ABR, A10, A11, A12,
A20, A21, A22;
FLA_Obj BT, B0,
BB, B1,
B2;
int b;
FLA_Part_2x2(A, &ATL, &ATR,
&ABL, &ABR, 0, 0, FLA_TL);
FLA_Part_2x1(B, &BT,
&BB, 0, FLA_TOP);
while (FLA_Obj_length(ATL) < FLA_Obj_length(A)){
b = min(FLA_Obj_length(ABR), nb_alg);
FLA_Repart_2x2_to_3x3(ATL, /**/ ATR, &A00, /**/ &A01, &A02,
/* ************* */ /* ****************** */
&A10, /**/ &A11, &A12,
ABL, /**/ ABR, &A20, /**/ &A21, &A22,
b, b, FLA_BR);
FLA_Repart_2x1_to_3x1(BT, &B0,
/* ** */ /* ** */
&B1,
BB, &B2, b, FLA_BOTTOM);
/*--*/
/* update line 1 */
/* : */
/* update line n */
/*--*/
FLA_Cont_with_3x3_to_2x2(&ATL, /**/ &ATR, A00, A01, /**/ A02,

Chol_blk_var3.c

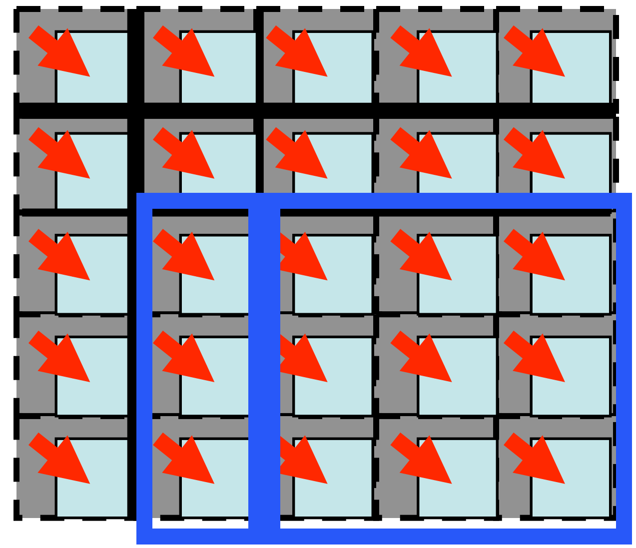
```c
int Syrk_ln_blk( FLA_Obj A, FLA_Obj B, int nb_alg )
{
  FLA_Obj ATL,   ATR,      A00, A01, A02,
          ABL,   ABR,      A10, A11, A12,
                           A20, A21, A22;

  FLA_Obj BT,              B0,
          BB,              B1,
                           B2;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  FLA_Part_2x1( B,    &BT,
                      &BB,            0, FLA_TOP );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           b, b, FLA_BR );

    FLA_Repart_2x1_to_3x1( BT,                &B0,
                        /* ** */            /* ** */
                                              &B1,
                           BB,                &B2,        b, FLA_BOTTOM );

    /*------------------------------------------------------------*/
```



1:** Chol_blk_var3.c   14% (12,37)   (C/l Abbrev)

Chol_blk_var3.c

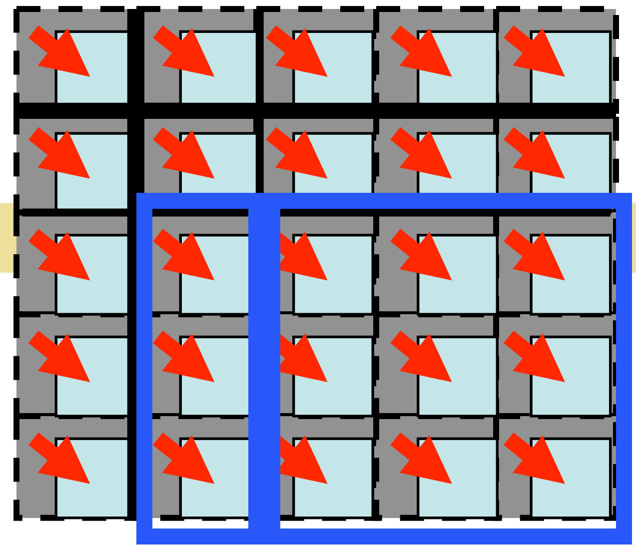
```
int Syrk_ln_blk( FLA_Obj A, FLA_Obj B )
{
  FLA_Obj ATL,    ATR,       A00, A01, A02,
          ABL,    ABR,       A10, A11, A12,
                             A20, A21, A22;

  FLA_Obj BT,                B0,
          BB,                B1,
                             B2;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  FLA_Part_2x1( B,    &BT,
                      &BB,            0, FLA_TOP );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){


    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           1, 1, FLA_BR );

    FLA_Repart_2x1_to_3x1( BT,                &B0,
                        /* ** */            /* ** */
                                              &B1,
                           BB,                &B2,        1, FLA_BOTTOM );

    /*------------------------------------------------------------*/
```



1:** Chol_blk_var3.c   14% (43,59)   (C/l Abbrev)

Chol_blk_var3.c

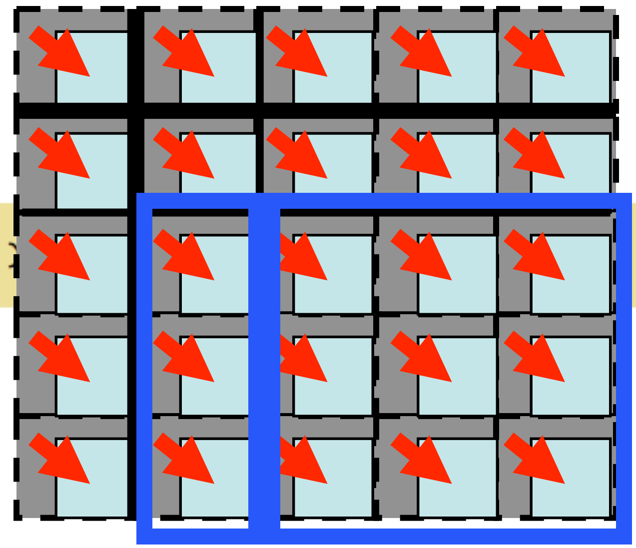
```c
                     /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                        ABL, /**/ ABR,          &A20, /**/ &A21, &A22,
                        1, 1, FLA_BR );

    FLA_Repart_2x1_to_3x1( BT,                &B0,
                     /* ** */              /* ** */
                                              &B1,
                        BB,                   &B2,        1, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    /*                      update line 1                       */
    /*                            :                             */
    /*                      update line n                       */

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                         /* ************** */  /* ***************** */
                           &ABL, /**/ &ABR,          A20, A21, /**/ A22,
                           FLA_TL );

    FLA_Cont_with_3x1_to_2x1( &BT,                  B0,
                                                    B1,
                         /* ** */              /* ** */
                           &BB,                     B2,     FLA_TOP );

  }

  return FLA_SUCCESS;
}
```



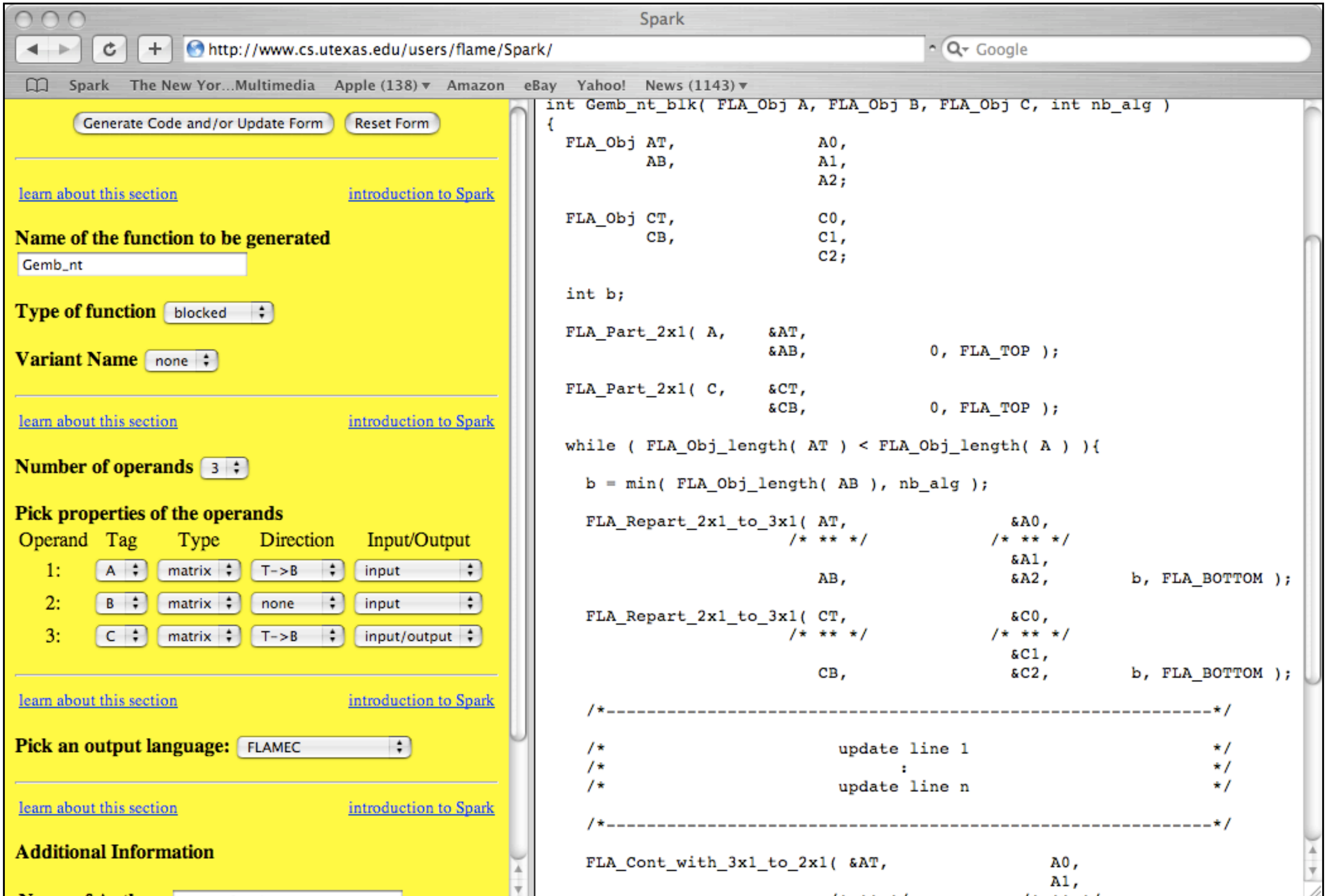
1:** Chol_blk_var3.c   Bot (47,0)   (C/l Abbrev)

Chol_blk_var3.c

```c
                          /* ************* */   /* ******************** */
                                                  &A10, /**/ &A11, &A12,
                             ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                             1, 1, FLA_BR );

    FLA_Repart_2x1_to_3x1( BT,                &B0,
                          /* ** */            /* ** */
                                              &B1,
                             BB,              &B2,        1, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE,
              FLA_MINUS_ONE, FLASH_MATRIX_AT( B1 ), FLA_ONE, FLASH_MATRIX_AT( A11 ) )
    FLASH_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE,
                FLA_MINUS_ONE, B2, B1, FLA_ONE, A21 );

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );

    FLA_Cont_with_3x1_to_2x1( &BT,                B0,
                                                  B1,
                            /* ** */            /* ** */
                              &BB,                B2,     FLA_TOP );

  }

  return FLA_SUCCESS;
}
```



1:** Chol_blk_var3.c   Bot (47,0)   (C/l Abbrev)


Spark
http://www.cs.utexas.edu/users/flame/Spark/
Google
Spark The New Yor...Multimedia Apple (138) Amazon eBay Yahoo! News (1143)
Generate Code and/or Update Form
Reset Form
learn about this section
introduction to Spark
Name of the function to be generated
Gemb_nt
Type of function
blocked
Variant Name
none
learn about this section
introduction to Spark
Number of operands
3
Pick properties of the operands
Operand Tag Type Direction Input/Output
1: A matrix T->B input
2: B matrix none input
3: C matrix T->B input/output
learn about this section
introduction to Spark
Pick an output language:
FLAMEC
learn about this section
introduction to Spark
Additional Information
int Gemb_nt_blk(FLA_Obj A, FLA_Obj B, FLA_Obj C, int nb_alg)
{
FLA_Obj AT, A0,
AB, A1,
A2;
FLA_Obj CT, C0,
CB, C1,
C2;
int b;
FLA_Part_2x1(A, &AT,
&AB, 0, FLA_TOP);
FLA_Part_2x1(C, &CT,
&CB, 0, FLA_TOP);
while (FLA_Obj_length(AT) < FLA_Obj_length(A)){
b = min(FLA_Obj_length(AB), nb_alg);
FLA_Repart_2x1_to_3x1(AT, &A0,
/* ** */ /* ** */
&A1,
AB, &A2, b, FLA_BOTTOM);
FLA_Repart_2x1_to_3x1(CT, &C0,
/* ** */ /* ** */
&C1,
CB, &C2, b, FLA_BOTTOM);
/*--*/
/* update line 1 */
/* : */
/* update line n */
/*--*/
FLA_Cont_with_3x1_to_2x1(&AT, A0,
A1,

Chol_blk_var3.c

```
int Gemb_nt_blk( FLA_Obj A, FLA_Obj B, FLA_Obj C, int nb_alg )
{
  FLA_Obj AT,              A0,
          AB,              A1,
                           A2;

  FLA_Obj CT,              C0,
          CB,              C1,
                           C2;

  int b;

  FLA_Part_2x1( A,    &AT,
                      &AB,            0, FLA_TOP );

  FLA_Part_2x1( C,    &CT,
                      &CB,            0, FLA_TOP );

  while ( FLA_Obj_length( AT ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( AB ), nb_alg );

    FLA_Repart_2x1_to_3x1( AT,                &A0,
                        /* ** */            /* ** */
                                              &A1,
                           AB,                &A2,        b, FLA_BOTTOM );

    FLA_Repart_2x1_to_3x1( CT,                &C0,
                        /* ** */            /* ** */
                                              &C1,
                           CB,                &C2,        b, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    /*                      update line 1                        */
```

1:** Chol_blk_var3.c   2% (3,60)   (C/l Abbrev)

Chol_blk_var3.c

```c
int Gemb_nt_blk( FLA_Obj A, FLA_Obj B, FLA_Obj C )
{
  FLA_Obj AT,              A0,
          AB,              A1,
                           A2;

  FLA_Obj CT,              C0,
          CB,              C1,
                           C2;

  int b;

  FLA_Part_2x1( A,    &AT,
                      &AB,            0, FLA_TOP );

  FLA_Part_2x1( C,    &CT,
                      &CB,            0, FLA_TOP );

  while ( FLA_Obj_length( AT ) < FLA_Obj_length( A ) ){



    FLA_Repart_2x1_to_3x1( AT,                &A0,
                        /* ** */            /* ** */
                                              &A1,
                           AB,                &A2,        1, FLA_BOTTOM );

    FLA_Repart_2x1_to_3x1( CT,                &C0,
                        /* ** */            /* ** */
                                              &C1,
                           CB,                &C2,        1, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    /*                          update line 1                     */
```

1:** Chol_blk_var3.c   2% (33,59)   (C/l Abbrev)

Chol_blk_var3.c

```
    FLA_Repart_2x1_to_3x1( AT,                &A0,
                        /* ** */            /* ** */
                                              &A1,
                           AB,                &A2,        1, FLA_BOTTOM );

    FLA_Repart_2x1_to_3x1( CT,                &C0,
                        /* ** */            /* ** */
                                              &C1,
                           CB,                &C2,        1, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    /*                       update line 1                        */
    /*                             :                              */
    /*                       update line n                        */

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x1_to_2x1( &AT,                A0,
                                                  A1,
                            /* ** */          /* ** */
                              &AB,                A2,     FLA_TOP );

    FLA_Cont_with_3x1_to_2x1( &CT,                C0,
                                                  C1,
                            /* ** */          /* ** */
                              &CB,                C2,     FLA_TOP );

  }

  return FLA_SUCCESS;
}
```

1:** Chol_blk_var3.c Bot (40,0) (C/l Abbrev)

Chol_blk_var3.c

```
    FLA_Repart_2x1_to_3x1( AT,                &A0,
                        /* ** */            /* ** */
                                              &A1,
                           AB,                &A2,        1, FLA_BOTTOM );

    FLA_Repart_2x1_to_3x1( CT,                &C0,
                        /* ** */            /* ** */
                                              &C1,
                           CB,                &C2,        1, FLA_BOTTOM );

    /*------------------------------------------------------------*/

    FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE,
              FLA_MINUS_ZERO, FLASH_MATRIX_AT( A1 ), FLASH_MATRIX_AT( B ),
              FLA_ONE, FLASH_MATRIX_AT( C1 ) );

    /*------------------------------------------------------------*/

    FLA_Cont_with_3x1_to_2x1( &AT,                A0,
                                                  A1,
                            /* ** */            /* ** */
                              &AB,                A2,     FLA_TOP );

    FLA_Cont_with_3x1_to_2x1( &CT,                C0,
                                                  C1,
                            /* ** */            /* ** */
                              &CB,                C2,     FLA_TOP );

  }

  return FLA_SUCCESS;
}
```

```
1:**  Chol_blk_var3.c    Bot (37,0)     (C/l Abbrev)
Auto-saving...done
```

Done!

Overview

- A bit of history
- Programming algorithms-by-blocks is ~~hard~~ easy
  - An example: Cholesky factorization by blocks
- Programming algorithms-by-blocks ~~further~~ simplifies ~~complicates~~ programming (future) multicore architectures
- Conclusion
- Resources

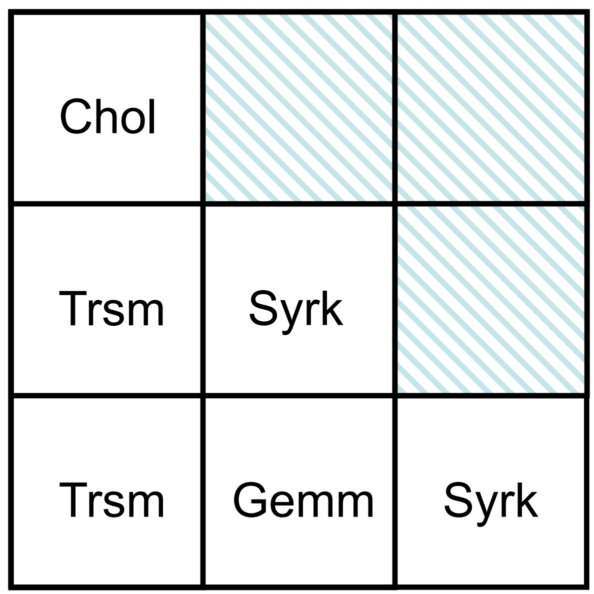
Programming Dense-Matrix on Multicore is ~~hard~~ easy

The basic idea

- Algorithms-by-blocks
  - Unit of data is a block
  - Unit of computation is a (BLAS-like) operation with blocks
  - Apply superscalar techniques at the block level
  - In software (runtime system) rather than hardware

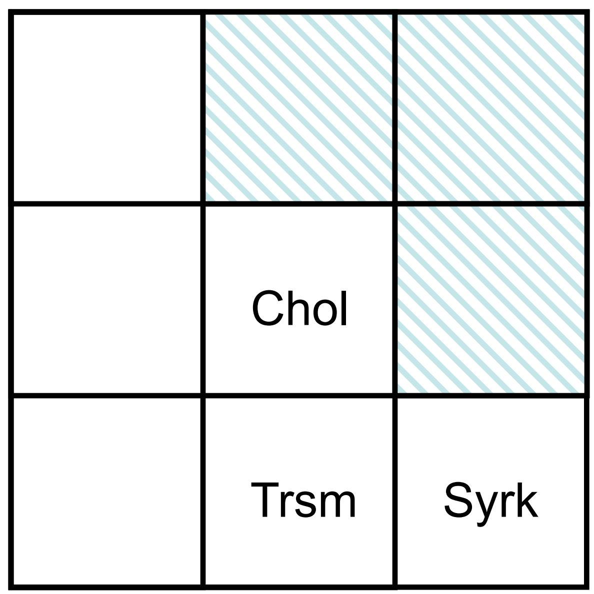
SuperMatrix

- Cholesky Factorization
  - Iteration 1



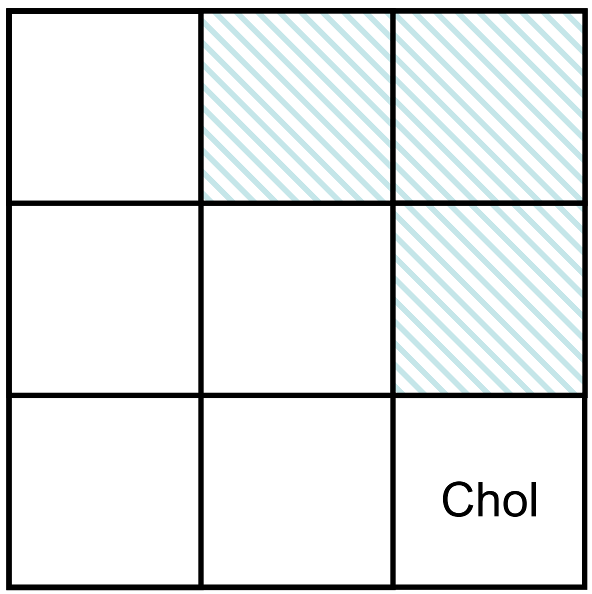
SuperMatrix

- Cholesky Factorization
  - Iteration 2



SuperMatrix

- Cholesky Factorization
  - Iteration 3



Basic idea (continued)

- Analyzer:
  - Build the DAG: Calls like
    `FLA_Chol(FLA_LOWER_TRIANGULAR,FLASH_MATRIX_AT(A11))`
    place a tasks on a queue
  - DAG: nodes are tasks, edges are dependencies.
- At run time check if dependencies have been met. Schedule tasks to threads.
- Akin to Tomasulo's algorithm and instruction-level parallelism on blocks of computation
- Runtime system: SuperMatrix

Chol_blk_var3.c

```
int Chol_blk_var3( FLA_Obj A )
{
  FLA_Obj ATL,   ATR,      A00, A01, A02,
          ABL,   ABR,      A10, A11, A12,
                           A20, A21, A22;

  int b;

  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           1, 1, FLA_BR );

    /*------------------------------------------------------------*/
    FLA_Chol_unb_var3( FLASH_MATRIX_AT( A11 ) );
    FLASH_Trsm( FLA_SIDE_RIGHT, FLA_LOWER_TRIANGULAR,
                FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
                FLA_ONE, A11, A21 );
    FLASH_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, FLA_MINUS_ONE,
                A21, FLA_ONE, A22 );
    /*------------------------------------------------------------*/
    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
```

1:** Chol_blk_var3.c   2% (27,16)   (C/l Abbrev)

The FLASH code generates the DAG!

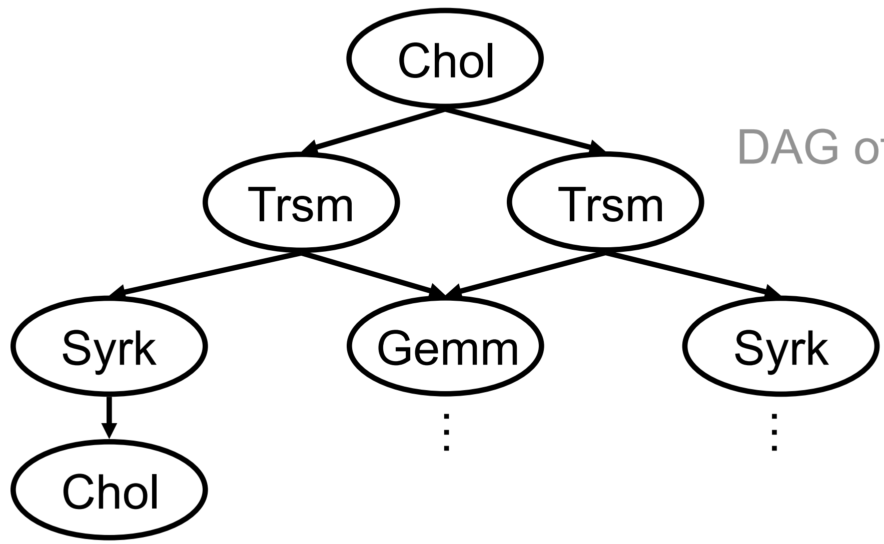
Completely transparent to library coder.

SuperMatrix

- Analyzer

Task Queue

Chol
Trsm
Trsm
Syrk
Gemm
Syrk
Chol
⋮



DAG of tasks

SuperMatrix

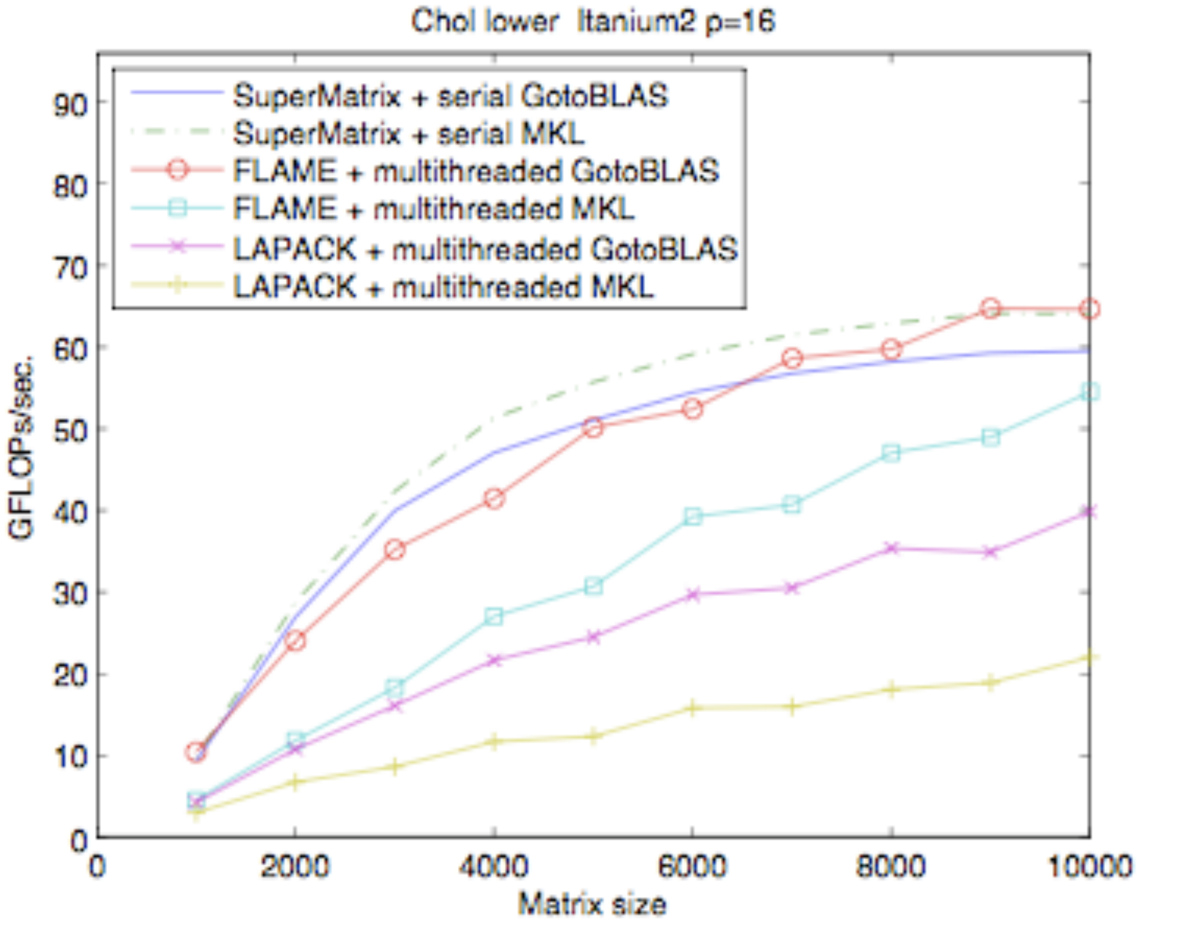
- Dispatcher
  - Use DAG to execute tasks out-of-order in parallel
  - Akin to Tomasulo's algorithm and instruction-level parallelism on blocks of computation
    - SuperScalar vs. SuperMatrix

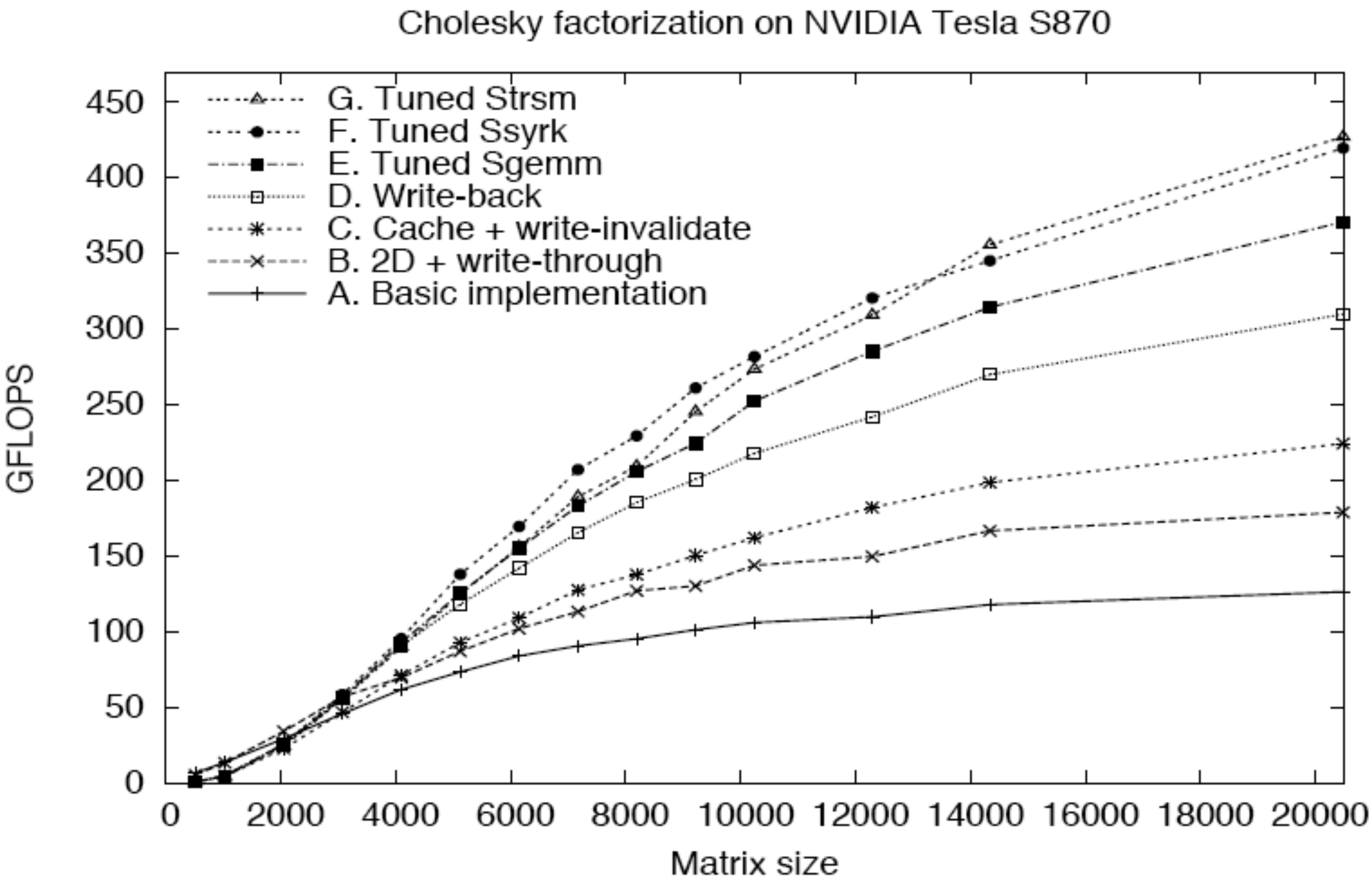
SuperMatrix

- Dispatcher
  - 4 threads
  - 5 x 5 matrix of blocks
  - 35 tasks
  - 14 stages

Chol			
Trsm	Trsm	Trsm	Trsm
Syrk	Gemm	Syrk	Gemm
Gemm	Syrk	Gemm	Gemm
Gemm	Syrk	Chol	
Trsm	Trsm	Trsm	
Syrk	Gemm	Syrk	Gemm
Gemm	Syrk	Chol	
Trsm	Trsm		
Syrk	Gemm	Syrk	
Chol			
Trsm			
Syrk			
Chol			

Performance




Cholesky factorization on NVIDIA Tesla S870
G. Tuned Strsm
F. Tuned Ssyrk
E. Tuned Sgemm
D. Write-back
C. Cache + write-invalidate
B. 2D + write-through
A. Basic implementation
GFLOPS
Matrix size
0
50
100
150
200
250
300
350
400
450
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000

Scope of Methodology

- Implemented so far:
  - Cholesky: A few weeks (includes writing the SuperMatrix runtime system)
  - BLAS3: 2 people, 1 weekend
  - LU-by-blocks with incremental pivoting: 1 person, a few days
  - QR factorization-by-blocks : 1 person, a few days
  - Solution of triangular Sylvester equation: 1 person, a few days

Overview

- A bit of history
- Programming algorithms-by-blocks is ~~hard~~ easy
  - An example: Cholesky factorization by blocks
- Programming algorithms-by-blocks ~~further~~ simplifies ~~complicates~~ programming (future) multicore architectures
- Conclusion
- Resources

Conclusion

- Is it time to stop evolving from LINPACK?
- Coding at a high level of abstraction yields a more flexible design
- The time for algorithms-by-blocks appears to have arrived
- Often new algorithms are required to support algorithms by blocks

Resources

The Science of Programming Matrix Computations

by

Robert A. van de Geijn
The University of Texas at Austin
and
Enrique S. Quintana-Ortí
Universidad Jaime I de Castellón

ONLY $8.34

at www.lulu.com

- www.cs.utexas.edu/users/flame
- www.linearalgebrawiki.org
- libFLAME R2.0 04.01.08